A Simple Algorithm for Consistent Query Answering under Primary Keys

Diego Figueira
Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI, UMR 5800, Talence, France

Anantha Padmanabha
DI ENS, ENS, CNRS, PSL University, Paris
Inria, France

Luc Segoufin
INRIA, ENS-Paris, PSL University

Cristina Sirangelo
Université Paris Cité, CNRS, Inria, IRIF, F-75013, Paris, France

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 database repairs with at most k facts, 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 $S' \in \Delta$ contained in $S \cup \{a\}$.

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.

2012 ACM Subject Classification Theory of computation → Database query languages (principles)

Keywords and phrases consistent query answering, primary keys, conjunctive queries

1 Introduction

A database often comes with integrity constraints. The 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 fulfilled. This task is not easy as it is inherently non deterministic: there may 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, postpone the problem until a query is asked to the database. In order to evaluate the query, the classical solution is to consider all possible repairs of the database and to output all the certain answers on the database D, *i.e.*, those answers that are in the output of the query when evaluated on *every* repair of D [2]. This method usually has a huge impact on the complexity of the query evaluation problem. The impact will of course depend on the type of integrity constraints and on the definition of a repair, but most often the worst case complexity increases by a

factor at least exponential in the size of the database, as there could be exponentially many ways to repair a database.

Depending on the integrity constraints, 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 relation *key*. An inconsistent database is therefore a database that has distinct tuples within a relation sharing the same key. 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 among all tuples having the same key in the same relation. A simple analysis shows that there can be exponentially many repairs for a given database, and therefore a naive evaluation algorithm would have to evaluate the query on each of these exponentially many repairs.

We consider Boolean conjunctive queries which can be evaluated efficiently over all databases 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 since, 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 answering problem is coNP-hard [3] while, for other queries, the certain answering problem can be solved in polynomial time. The main conjecture for inconsistent databases in the presence of primary keys is that there are no intermediate cases: for any conjunctive query, the certain answering problem is either solvable in polynomial time or is complete for coNP.

The conjecture has been proved for self-join-free Boolean conjunctive queries [7]. These are queries on which there are no two atoms using the same relation. It has been also proved for path queries [6]. 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.

Contributions Our main contribution is the design of a simple fixpoint algorithm for computing certain answers of 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: when it outputs 'yes' then the query is certain, *i.e.*, it is true on all repairs of the database. But there could false negatives: certain queries on which the algorithm outputs 'no'.

Our main result shows that this simple algorithm is correct for all self-join-free queries and path queries whose certain answers problem is computable in polynomial time. In other words, if for all k our algorithm gives a false negative answer for a self-join-free or path query it is because the query has a coNP-complete certain answering problem.

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. We answer negatively to this question, by exhibiting a conjunctive query (with self-joins) whose certain answers can be solved in polynomial time, but for all k the algorithm fails to give a correct answer.

Though our greedy fixpoint computation algorithm is simple, the proof of correctness is

not. In the case of self-join-free queries, we provide a semantic condition and show that when the condition holds, the fixpoint gives always the correct answer, by setting the parameter k to be the number of atoms of the query. The proof is by contradiction: if the algorithm fails to give the correct answer, we use the fixpoint definition of the algorithm in order to produce (chase) an infinite sequence of distinct facts of the database, contradicting its finiteness. When the semantic condition does not hold, we show that it implies the condition of [7] characterizing those queries having a coNP-complete certain answers problem.

The situation is simpler for the case of path queries, as we show that for a suitable k (again the number of atoms of the query), our fixpoint algorithm can simulate the polynomial time algorithm of [6] for computing certain answers for q, assuming that certain answering for q is polynomial time solvable.

Related work Our work is definitely inspired by the results of Koutris and Wijsen [8, 7]. In the self-join-free case their proof of the polynomial case is a long sequence of reductions which eventually produces 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 probably as complex as theirs. Further, our algorithm does not give, a priori, the optimal LogSpace complexity result of [8] as we know that some of the path queries that can be solved with our algorithm are PTime complete [6]. 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 [7] based on the so-called "attack graph" of the query.

In the case of path queries, [6] 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.

2 Preliminaries

A **database** is a finite relational structure. 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 of D, and a function associating to each symbol R of σ a relation $R(D)$ of the appropriate arity over the domain of D.

An **R-fact** of a database D over a 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** is an R-fact for some R, R is then the symbol associated to the fact and $\bar{a}$ the tuple associated to the fact. A database can then be viewed as a finite collection of facts. By the **size of a database** we mean 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 **primary key constraint** over a signature σ is a special case of a functional dependency designating for a relation symbol R of σ a certain set of indices (columns) of R as a primary key. A database satisfies the primary key constraint if for every relation R over σ, whenever two R-facts agree on the key indices they must be equal. In a set of primary key constraints, each relation of σ has a unique primary key constraint. 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 $u \sim_\Gamma v$, 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 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 describing explicitly Γ. For instance $R(\underline{x}\ y)$ says that the position of the variable x (*i.e.*, the first position) is the key for the binary relational symbol R; and $R'(\underline{yz}\ x)$ says that the positions of the variables y and z form the key for the ternary relational symbol R'.

If a database D satisfies the key constraints Γ, denoted $D \models \Gamma$, 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.

In this paper a query is a Boolean conjunctive query. We view a query over a relational signature σ as a collection of atoms where an atom is a term $R(\bar{x})$ where R is a relation symbol 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 be ordered in an arbitrary but fixed order. 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 $A_{i_1}, \ldots, A_{i_l}$.

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 $\mathbf{certain}_\Gamma(q)$ (or simply 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 CERTAIN(q) is coNP-complete [3]. However, there are queries q for which CERTAIN(q) is in PTime or even expressible in FO [5, 9].

The following dichotomy has been conjectured (cf [3, 1]):

▶ **Conjecture 1** (Dichotomy conjecture). *For each query q, the problem* CERTAIN(q) *is either in* PTime *or* coNP*-complete.*

The conjecture has been confirmed in the case of self-join-free queries [7] and of path queries [6]; however, it remains open in the general case. A Boolean conjunctive query is **self-join-free** if all its atoms involve different relational symbols. 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 $A_i = R_i(\underline{x_{i-1}}, x_i)$ for some symbol R_i of σ of arity two. The query may contain self-joins, i.e. $R_i = R_j$ for some $i \neq j$.

▶ **Example 2.** Consider the following example queries taken from [5, 6]. For the self-join-free query $q_1 : R_1(\underline{x}\ y) \wedge R_2(\underline{y}\ z)$ (recall that all variables are implicitly existentially quantified), it is easy to see that the problem CERTAIN(q_1) can be solved in polynomial time [5]. Actually, it can be expressed by the first-order formula $\exists xyz\ R_1(xy) \wedge R_2(yz) \wedge \forall y'(R_1(xy') \rightarrow \exists z' R_2(y'z'))$.

For the self-join-free query $q_2 : R_1(\underline{x}\ y) \wedge R_2(\underline{y}\ x)$ and the path query $q_2' : R(\underline{x_1}\ x_2) \wedge X(\underline{x_2}\ x_3) \wedge R(\underline{x_3}\ x_4) \wedge Y(\underline{x_4}\ x_5) \wedge R(\underline{x_5}\ x_6) \wedge Y(\underline{x_6}\ x_7)$, it has been shown, in [9] and [6] respectively, that CERTAIN(q_2) and CERTAIN(q_2') can be solved in polynomial time but cannot be expressed in first-order logic. Our algorithm works for both q_2 and q_2'.

Finally, for the self-join-free query $q_3 : R_1(\underline{x}\ y) \wedge R_2(\underline{z}\ y)$ and the path query $q_3' : R(\underline{x_1}\ x_2) \wedge X(\underline{x_2}\ x_3) \wedge R(\underline{x_3}\ x_4) \wedge X(\underline{x_4}\ x_5) \wedge R(\underline{x_5}\ x_6) \wedge Y(\underline{x_6}\ x_7) \wedge R(\underline{x_7}\ x_8) \wedge Y(\underline{x_8}\ x_9)$, both CERTAIN($q_3$) and CERTAIN($q_3'$) are known to be coNP-complete [3, 6].

3 Polynomial-time algorithm

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

In Section 4 we will show that for self-join-free queries either Cert$_k(q)$ computes CERTAIN(q) (where k is the number of atoms occurring in q) or CERTAIN(q) is complete for coNP. In Section 5 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 such that no two elements of S are Γ-equivalent. In other words a k-set is a subset of a repair of D of size at most k. We denote by Cert$_k(q)$ the following algorithm. On a database input D, the algorithm Cert$_k(q)$ computes inductively a set $\Delta_k(q, D)$ of k-sets over D while maintaining the invariant:

If $S \in \Delta_k(q, D)$ and r is a repair of D containing S, 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 $u \in B$ 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 inductive steps can be expressed in FO. If n is the number of facts of D, the fixpoint is reached in at most n^k steps. In the end, Cert$_k(q)$ returns 'yes' iff the empty set belongs to $\Delta_k(q, D)$. Equivalently, 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 $D \models \mathbf{Cert}_k(q)$ to denote the fact that Cert$_k(q)$ returns "true" upon input D. Altogether we have shown:

▶ **Proposition 3.** *For all q, Γ, k, Cert$_k(q)$ runs in time polynomial in the size of its input database D and, whenever $D \models$ Cert$_k(q)$ then q is certain for D.*

In order to simplify the notations, as we will mostly consider this case, we write $\Delta(q, D)$ and Cert(q) to denote $\Delta_k(q, D)$ and Cert$_k(q)$ respectively, where k is the number of atoms of q. Also, for a fact u, we write $u \in \Delta_k(q, D)$ instead of $\{u\} \in \Delta_k(q, D)$.

▶ **Example 4.** Consider again the query $q_2 : R_1(\underline{x}\ y) \wedge R_2(\underline{y}\ x)$ from Example 2. Let $k = 2$ and consider the execution of Cert$_2(q_2)$. Initially, $\Delta_2(q_2, D)$ 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 $\Delta_2(q_2, D)$ (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. And so it continues.

As a teaser to the next section, the reader can see why $\mathrm{Cert}_2(q_2)$ computes $\textsc{certain}(q_2)$ in Appendix A. This is not always true as for the query q_3 of Example 2, $\textsc{certain}(q_3)$ being coNP-complete, $\mathrm{Cert}_k(q_3)$ must have false negatives for all k (assuming coNP $\neq$ PTime).

4 Self-join-free queries

In this section we consider the case of self-join-free queries. We exhibit a condition named PH (for Polynomial-time Hypothesis) and show that any self-join-free query q satisfying PH is such that $\mathrm{Cert}(q)$ computes $\textsc{certain}(q)$. When PH fails, we show that $\textsc{certain}(q)$ is coNP-hard.

We start by defining PH, which will require some extra definitions. Fix, for the rest of this section, a set Γ of key constraints. Let D be a database and r a repair of D. For a fact u of r and $v \sim u$ be an equivalent fact 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 $\bar{u}$ and on the atoms of q is 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 same solution, placing up front the most relevant facts. In particular we shall write, for a tuple $\bar{u}$ of facts, $\bar{u} \in \Delta(q, D)$ to denote that the k-set formed with the facts of $\bar{u}$ belongs to $\Delta(q, D)$.

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

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

If $X = vars(A_0)$, for some atom A_0 of q, we also say that the Γ-derivation is from A_0 to A_n, and we also 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 said ***stable*** if each pair of facts of S are mutually Γ-determined. Note that a stable set is not necessarily an equivalence class of Γ-determinacy, it may also be a subset of it. As usual, we will omit Γ when it is clear from the context. The key property relating Γ-determinacy and query solutions is given by the following lemma and its corollaries :

▶ **Lemma 5.** *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. For $n = 1$ the statement trivially holds since $key(A_1) \subseteq X$ thus $\mu(A_1) \sim \mu'(A_1)$.

Now consider a sequence $XA_1 \dots A_n$, $n > 1$ satisfying the hypotheses. The induction hypothesis applied to the sequence $XA_1 \dots A_{n-1}$ implies $\mu(A_i) = \mu'(A_i)$, for all $1 \leq i \leq n-1$. Then μ and μ' agree on $vars(A_1) \cup \dots \cup vars(A_{n-1})$.

Since $XA_1 \dots A_n$ is a Γ-derivation sequence, we have $key(A_n) \subseteq X \cup vars(A_1) \cup \dots \cup vars(A_{n-1})$; hence μ and μ' agree on $key(A_n)$, or in other words $\mu(A_n) \sim \mu'(A_n)$. ◀

▶ **Corollary 6.** *Let q be a self-join-free query. Let D be a database instance and r, r' be two repairs of D. Let $A_1 \dots A_n$ be a Γ-derivation in q from atom A_1 to A_n. Let $r \models q(\bar{\alpha} a_1 \dots a_n \bar{\beta})$, and $r' \models q(\bar{\alpha}' a'_1 \dots a'_n \bar{\beta}')$ where for each i, a_i and a'_i match A_i and $\bar{\alpha}\bar{\beta}$ and $\bar{\alpha}'\bar{\beta}'$ matches the rest of the atoms of q. If $a_1 = a'_1$ and $a_i \in r'$ for each $i < n$, then 1) $a_i = a'_i$ for each $i < n$ and 2) $a_n \sim a'_n$ (and therefore $a_n = a'_n$ if moreover $a_n \in r'$).*

Proof of the corollary. For $n = 1$ the statement is trivial. For $n > 1$ it follows directly from Lemma 5 using the Γ-derivation $vars(A_1)\ A_2 \dots A_n$. ◀

▶ **Corollary 7.** *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 of the corollary. The statement follows directly from Corollary 6 using $r = r'$, with a_1 being any fact in $\bar{a} \cap \bar{a}'$, A_1 being the atom matched by 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 PH. 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 $\bigcup_{j \leq i} S_j$. 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 $\mathbf{PH}_\tau(A)$ and write $q \models \mathrm{PH}_\tau(A)$ 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:

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

We write $q \models \mathbf{PH}_\tau(i)$ if q satisfies $\mathrm{PH}_\tau(A)$ for all A of S_{i+1} and $q \models \mathbf{PH}_\tau$ if q satisfies $\mathrm{PH}_\tau(i)$ for all $1 \leq i < n$. Since the condition is restricted to indices $i < n$, PH_τ trivially holds for any τ having only one stable set. Finally, we write $q \models \mathbf{PH}$ if there is a Γ-sequence τ of q such that $q \models \mathrm{PH}_\tau$. Again, if q has only one Γ-determinacy class, then $q \models \mathrm{PH}$ in a trivial way.

Our goal is to show that $q \models \mathrm{PH}$ implies that $\mathrm{Cert}(q)$ computes CERTAIN(q). This is the main technical result of this section and is proved by Theorem 9. In Theorem 15 we will conclude the self-join-free case by showing that when PH fails, then the certainty of the query is hard.

Before we prove those results, some examples are in order.

▶ **Example 8.** We recall the three queries from Example 2.

The query $q_2 = R_1(\underline{x}\ y) \wedge R_2(\underline{y}\ x)$ satisfies PH since it has only one maximal stable set.

The query $q_1 = R_1(\underline{x}\ y) \wedge R_2(\underline{y}\ z)$ 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 $q \not\models \mathrm{PH}_\tau$ because we have $q_1(R_2(bc)R_1(ab))$ and $q_1(R_2(b'c)R_1(ab'))$ but not $q_1(R_2(bc)R_1(ab'))$. However for $\tau = \{R_1(\underline{x}\ y)\}\{R_2(\underline{y}\ z)\}$ it is easy to verify that $q_1 \models \mathrm{PH}_\tau$. Hence, $q_1 \models \mathrm{PH}$.

The query $q_3 = R_1(\underline{x}\ y) \wedge R_2(\underline{z}\ y)$ has also two stable sets, but no possible sequence τ makes PH_τ true. This is because (i) $q_3(R_1(ab)\ R_2(cb))$ and $q_3(R_1(a'b')\ R_2(cb'))$ hold, but not $q_3(R_1(ab)\ R_2(cb'))$, and (ii) $q_3(R_2(ab)\ R_1(cb))$ and $q_3(R_2(a'b')\ R_1(cb'))$ hold, but not $q_3(R_2(ab)\ R_1(cb'))$. Therefore, $q_3 \not\models \mathrm{PH}$.

▶ **Theorem 9.** *Let q be a self-join-free query. If $q \models \mathrm{PH}$, then $\mathrm{Cert}(q)$ computes* CERTAIN(q).

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

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 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 S_0 is the empty set, as well as $S_{\leq 0}$ (and hence $\mathrm{PH}_\tau(0)$ 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 will make use of an induction based on the following invariant property of the database, for each $0 \leq i \leq n$:

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

▶ **Lemma 10.** *Given q, D and a Γ-sequence τ for q, for every $0 \leq i < n$, if* $\textsc{Ind}_{i+1}$ *and* $\mathrm{PH}_\tau(i)$*, then* $\textsc{Ind}_i$*.*

We first show how this statement already implies Theorem 9.

Proof of Theorem 9. From Proposition 3, we know that if D is a database such that $D \models \mathrm{Cert}(q)$ then all repairs of D satisfy q. It remains to show the converse.

Assume all repairs satisfy q and that $q \models \mathrm{PH}_\tau$ for some sequence τ of length n, which means that $\mathrm{PH}_\tau(i)$ for all i. Observe that $\textsc{Ind}_n$ holds true by the base case definition of $\Delta(q, D)$. Hence by n repeated applications of Lemma 10 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 $\Delta(q, D)$, and hence $D \models \mathrm{Cert}(q)$. ◀

We are now left with the proof of Lemma 10, 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** 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 a strong i-minimal repair is in particular $(i+1)$-minimal.

▶ **Claim 11.** *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 $s' \models q_{\leq i}(\bar{u})$.*

The claim is proved in the Appendix. 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 10. By contradiction, suppose the lemma is false. Then, there is some i such that IND_{i+1} and $\text{PH}_\tau(i)$ holds, but for some i-minimal repair s and tuple $\bar{u}$ we have

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

From Claim 11, 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 $r_0 = s$ and $\bar{p}_0$ to be the empty fact sequence. And for all $l > 0$, we define $\bar{p}_l = p_1, \ldots p_l$.

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) $\bar{u}\bar{c} \notin \Delta(q, D)$, 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 $r_{|i+1}$.

Note that the invariant (b) leads to a contradiction when l is larger than the size of D.

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 $\bar{u} \notin \Delta(q, D)$ by (h1).

Inductive step Assume we have $r = r_{l-1}$ and $\bar{p} = p_1, \ldots p_{l-1}$ (possibly empty) satisfying the five properties above. 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:

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

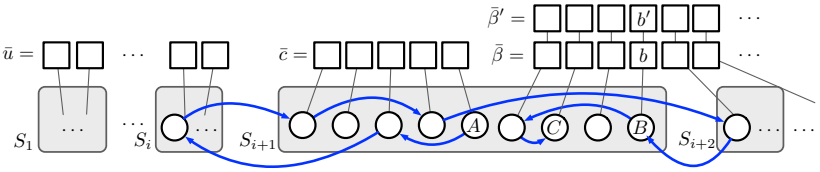
First let $\bar{\beta} = d_1 d_2, \ldots d_t$. Since $\bar{u}\bar{c} \notin \Delta(q, D)$, by definition of $\Delta(q, D)$ there exists some $d'_1 \sim d_1$ such that $\bar{u}\bar{c}d'_1 \notin \Delta(q, D)$. This again implies that there exists some $d'_2 \sim d_2$ such that $\bar{u}\bar{c}d'_1 d'_2 \notin \Delta(q, D)$. 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 $\bar{u}\bar{c}\beta'$ contains no k-set in $\Delta(q, D)$. Also note that $\bar{\beta}'$ matches all atoms of q not already matched by $\bar{u}\bar{c}$.

Further, we 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, since IND_{i+1} holds by hypothesis, $\bar{u}\bar{c} \in \Delta(q, D)$, which is in contradiction with (h2). Then,

$$r' \not\models q_{\leq i+1}(\bar{u}\bar{c}). \tag{h3}$$

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 fact matching B in $\bar{\beta}'$. We show that

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

Suppose b is in $\bar{p}$, by connectedness of $\bar{p}$ with respect to $r_{|i+1}$, this implies that the suffix $\bar{b}$ of $\bar{p}$ starting with b is part of the solution $\bar{u}\bar{c}\bar{\beta}$, that is, $r' \models q(\bar{u}\bar{b}\bar{\gamma})$ for some $\bar{\gamma}$. By construction, b is not in $\bar{c}$, thus it must occur before $\bar{c}$ in $\bar{p}$ and hence $\bar{b}$ strictly contains $\bar{c}$. 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. (Notice that, since p is not empty $A \neq B$.) 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 Corollary 6 to r' and t with solutions $(\bar{u}\bar{c}\bar{\beta})$ and $(\bar{\gamma}a\bar{\delta})$ respectively, and Γ-derivation σ_{AB}. The hypotheses of Corollary 6 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 Corollary 6, 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 (proved in the Appendix), 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 12.** *Assume* $\mathrm{PH}_\tau(i)$. *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 $\bar{\delta}$.*

(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 $\bar{c}$, $\bar{d}$ 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 $\Delta(q, D)$, we have $\bar{u}\bar{d}b' \notin \Delta(q, D)$ as needed.

This completes the proof of Lemma 10. ◀

▶ Remark 13. One can verify from the proof that if $q \models \mathrm{PH}_\tau$ where in the sequence τ each set S_i contains exactly one atom of q, then the fixpoint computing $\mathrm{Cert}(q)$ is bounded, *i.e.*, $\mathrm{Cert}(q)$ can be expressed in FO.

It remains to show that when a self-join-free query q does not satisfy PH, computing CERTAIN(q) is coNP-hard. Towards this, we build on the dichotomy result of [7] based on the notion of 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

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

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.

The dichotomy result of [7] can be stated as:

▶ **Theorem 14.** *[7, 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* CERTAIN(q) *can be computed in polynomial time; otherwise* CERTAIN(q) *is* coNP*-complete.*

To show that our polynomial time algorithm covers all polynomial cases, we prove that if the attack graph of q and Γ contains only weak cycles, then PH holds (proof in Appendix B).

▶ **Theorem 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 $q \models \mathrm{PH}$.*

5 Path queries

The dichotomy conjecture has also been shown to hold for *path queries* [6]. Recall that a path query of length n 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 $A_i = R_i(\underline{x_{i-1}}\ x_i)$ for some symbol R_i of σ. The query may contain self-joins, *i.e.*, there could be $R_i = R_j$ for $i \neq j$.

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 a primary key. Note that a path query can be described by a word over σ (*e.g.*, the word describing q is $R_1 \cdots R_n \in \sigma^*$). For simplicity, we will henceforth blur the distinction between path queries and words over σ.

Following [6] we define the language $\mathcal{L}^{\curvearrowright}(q)$ as the regular language defined by the following finite state automaton $\mathcal{A}^q$ with epsilon-transitions (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.

The dichotomy result of [6] can be formulated as follows[1]:

▶ **Theorem 16.** *[6, Theorem 3.2] Let q be a path query. If q is a factor of all the words in the language $\mathcal{L}^{\curvearrowright}(q)$, then* CERTAIN$(q)$ *can be evaluated in* PTIME*; otherwise,* CERTAIN(q) *is* CONP*-complete.*

We will show that, also in this case, Cert(q) also captures CERTAIN(q) for all polynomial-time path queries q, where $n = |q|$.

▶ **Theorem 17.** *Let q be a path query of length n. If q is a factor of all the words in the language $\mathcal{L}^{\curvearrowright}(q)$, then* CERTAIN$(q)$ = Cert(q).

The rest of this section is devoted to the proof of Theorem 17.

We will make use of the following fixpoint computation introduced by [6, Fig. 5]. For a fixed path query q and database instance D, let $N(q, D)$ be the set of pairs $\langle c, s\rangle$, where $c \in adom(D)$ and s is a prefix of q, computed via the following fixpoint algorithm.

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

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 $c \in adom(D)$ with $\langle c, \varepsilon\rangle \in N(q, D)$.

▶ **Lemma 18.** *[6, (proof of) Lemma 6.4] For every path query q, if q is a factor of every word in the language $\mathcal{L}^{\curvearrowright}(q)$, then* CERTAIN$(q) = N(q)$.

In view of Lemma 18, Theorem 17 is now a direct consequence of the following proposition.

[1] Actually, [6] provides a much finer tetrachotomy between FO, NL-complete, PTIME-complete and CONP-complete. Here we restrict our attention to the dichotomy between PTIME and CONP-complete.

▶ **Proposition 19.** *For every path query q of length n, and assuming every word of $\mathcal{L}^{\leftrightarrow}(q)$ contains q as factor, we have $N(q) = \mathrm{Cert}(q)$.*

Note that $\mathrm{Cert}(q) \subseteq N(q)$ follows from $\mathrm{Cert}(q) \subseteq \textsc{certain}(q)$ (Proposition 3) combined with $\textsc{certain}(q) = N(q)$ (Lemma 18). So we are left with proving $N(q) \subseteq \mathrm{Cert}(q)$. Let $D \in N(q)$. We will prove that $D \in \mathrm{Cert}(q)$.

For all $l \geq 0$ let s_l be the prefix of q of length l (*i.e.*, $s_0 = \varepsilon$ and $s_n = q$). For every database D and fact $u = R(\underline{a}\ b)$ in D, let us define $\mathit{trace}(u) = R$, $\mathit{key}(u) = a$, and $\mathit{last}(u) = b$. For a sequence of (possibly repeating) facts $\Pi = u_1, \ldots, u_k$ of a database D, we define $\mathit{last}(\Pi) = \mathit{last}(u_k)$ and $\mathit{trace}(\Pi) = \mathit{trace}(u_1) \cdots \mathit{trace}(u_k) \in \Sigma_q^*$. Also, let $S_\Pi = \{u_1, \ldots, u_k\}$ be the *set* of facts in the sequence. Further, Π is called a **valid path** if (i) S_Π is a partial repair of D, (ii) for all $i < k$ 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 k, we have $\mathit{trace}(\Pi) = s_k$. Further, for any prefix s_l of q we write $\boldsymbol{\mathit{trace}}(\Pi) \sim s_l$ if there exists a run of the automaton $\mathcal{A}^q$ on $\mathit{trace}(\Pi)$ ending in state s_l.

Let $D \in N(q)$. Let $N^i(q, D)$ and $\Delta^i(q, D)$ be the fix-point computations of $N(q, D)$ and $\Delta(q, D)$ at their i^{th} steps, respectively. To prove the Proposition $D \in \mathrm{Cert}(q)$, we will use the following claim:

▶ **Claim 20.** *For all $i \geq 0$, For $c \in \mathit{adom}(D)$ and for all prefix s_l of q if $\langle c, s_l\rangle \in N^i(q, D)$ then for all valid path Π where $\mathit{last}(\Pi) = c$ and $\mathit{trace}(\Pi) \sim s_l$ we have $S_\Pi \in \Delta^i(q, D)$.*

Let us show that the claim implies $D \in \mathrm{Cert}(q)$. As $D \in N(q)$, there exists $c \in \mathit{adom}(D)$ such that $\langle c, \varepsilon\rangle \in N^m(q, D)$ for some step m. But note that $\langle c, \varepsilon\rangle \in N^m(q, D)$ 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 $\langle b, R\rangle \in N^{m-1}(q, D)$. For each such b we can apply the claim with the valid path $\Pi = R(\underline{c}\ b)$, obtaining $\{R(\underline{c}\ b)\} \in \Delta^{m-1}(q, D)$ for every $R(\underline{c}\ b)$. Hence, $\emptyset \in \Delta^m(q, D)$ which implies $D \in \mathrm{Cert}(q)$.

The proof of Claim 20 is provided in the appendix.

6 Cert_k does not capture all polynomial instances

We have shown that for all self-join free and path queries whose certainty is solvable in polynomial time, there is a k such that Cert_k computes all certain answers. A natural question is whether this extends to all queries for which certainty is solvable in polynomial time. In this section, we show that this is not the case.

▶ **Theorem 21.** *Let q be the query $q = R(\underline{x}\, yz) \wedge R(\underline{z}\, xy)$. Then $\textsc{certain}(q)$ is in polynomial time but cannot be computed by $\mathrm{Cert}_k(q)$, for any k.*

We prove the theorem in the rest of the section. Since q contains only two atoms, for a database D, it is convenient to describe the set of solutions to q as a graph. More precisely we say that the facts $a, b \in D$ are *q-related* if $D \models q(ab)$ or $D \models q(ba)$. We define the solution graph of D, denoted by G_D to be an undirected graph whose vertices are the facts of D and pairs of q-related facts form the edges.

Note that q has a very rigid behavior. For every fact a in the database there is at most one fact b such that $q(ab)$ and at most one fact c such that $q(ca)$. Hence, the degree of every

node in G_D is bounded by 2. Also, for all $a, b, c \in D$ if $q(ab)$ and $q(bc)$ then $q(ca)$. Thus, every connected component in G_D is always a clique, of size at most 3. If a clique has exactly three vertices we call it a *triangle*.

First we show that CERTAIN(q) is in polynomial time.

▶ **Lemma 22.** CERTAIN(q) *is in* PTIME.

Proof. Fix an input database D. Without loss of generality, assume that there are no facts $a \in D$ such that $q(aa)$ is a solution[2]. We reduce CERTAIN(q, D) to a bipartite graph matching problem. Consider the bipartite graph $G = (V_1 \cup V_2, E)$ where V_1 is the set of all blocks and V_2 is the set of maximal cliques in 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 $v_1 \in V_1$. Construct repair r where for every block B of D, pick the fact (or any fact, should there be 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 $q(aa)$ in D, no two chosen facts will be q-related. Thus $r \not\models q$.

Conversely, if q is not certain in D, let r be a repair such that $r \not\models q$. 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 contains at least two elements of r; these two elements are therefore q-related, and thus $r \models q$.

Thus, to check if $D \in$ CERTAIN(q), it is sufficient to check if there is a bipartite matching of G that saturates V_1. This is known to be in PTIME [4]. ◀

We now prove that for all k, Cert$_k(q)$ does not compute CERTAIN(q). To this end, for every $n \geq 4$ we exhibit a database D_n such that

- $D_n \models$ CERTAIN(q) (Proposition 23);
- $D_n \not\models$ Cert$_{n-2}(q)$ (Proposition 24).

Fix some $n \geq 4$. The database D_n has n blocks of the form $B_1, \cdots, B_n$ where each B_i consists of $n-1$ facts denoted $b_i^1, \cdots, b_i^{n-1}$. Further, D_n also has $(n-1)(n-3)$ blocks of the form E_l^j for every $1 \leq j \leq n-1$ and $1 \leq l \leq n-3$, where each E_l^j has two facts denoted by u_l^j and v_l^j. The solution graph of D_n is depicted in Figure 1, and the formal definition of D_n can be found in the appendix.

D_n is defined in such a way that every fact of D_n is part of a "triangle". More precisely, we have the following triangles in the solution graph of D_n: For every $1 \leq j \leq n-1$, the triples $\{b_1^j, b_2^j, u_1^j\}$ and $\{b_{n-1}^j, b_n^j, v_{n-3}^j\}$ form triangles and for every $1 \leq l < n-3$ we have a triangle $\{v_l^j, u_{l+1}^j, b_{l+2}^j\}$.

▶ **Proposition 23.** *For every $n \geq 2$, $D_n \models$* CERTAIN(q).

To see this, note that there are $n + (n-1)(n-3)$ blocks in D_n and there are $(n-1)(n-2)$ cliques (all triangles) in the solution graph of D_n. Thus, in the corresponding bipartite graph

[2] If there is such a fact $a \in D$ then suppose the block containing a is a singleton set then clearly $D \models$ CERTAIN(q). Otherwise, $D \models$ CERTAIN(q) iff $D \setminus \{a\} \models$ CERTAIN(q), so we can consider the smaller database instance.

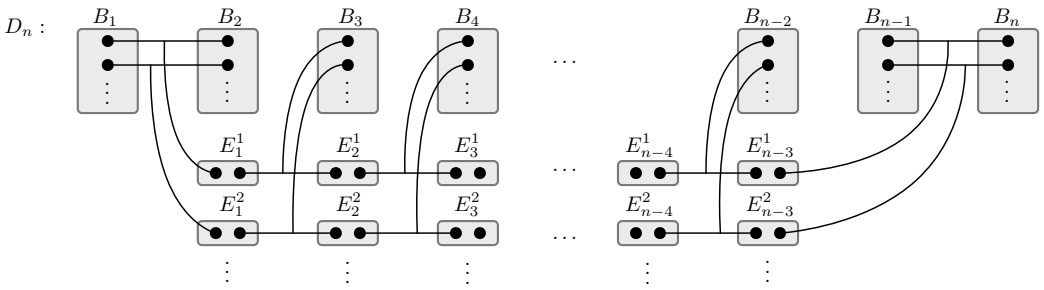


Figure 1 Solution graph for database D_n. Black discs denote facts, rectangles denote blocs, and three-pointed edges denote 'triangles' in the solution graph of D_n. There are $n-1$ facts in each B_i and two facts in every E_k^j.

(*cf.* Lemma 22) size of V_1 is strictly smaller than the size of V_2. Hence, there cannot be a V_1-saturating matching which implies $D_n \models \textsc{certain}(q)$.

▶ **Proposition 24.** *Let $k \geq 2$. $D_{k+2} \not\models \mathrm{Cert}_k(q)$.*

Proof sketch. To prove the proposition, for a set of blocks $\mathbb{X} = \{X_1, \ldots X_k\}$ of D_{k+2} if $W = \{a_1, \ldots a_k\}$ is a set of facts where each $a_i \in X_i$ then we call W, a partial repair of $\mathbb{X}$. Further, W is called an **obstruction set** of $\mathbb{X}$ if W satisfies some 'desired properties'. In particular we will show that if W is an obstruction set then $W \not\models q$. Next we will prove that for any set of blocks $\mathbb{X} = \{X_1, \ldots X_k\}$ of D_{k+2}, we can always pick a partial repair W of $\mathbb{X}$ such that W is an obstruction set.

Now suppose $D_{k+2} \in \mathrm{Cert}_k(q)$ then it follows that there has to exist at least one obstruction set $W \in \Delta_k(q, D_{k+2})$. We pick the minimum i such that there is some obstruction set in i^{th} step of $\Delta_k(q, D_{k+2})$ computation. Note that $i = 0$ is not possible since obstruction sets do not contain a solution to q. Thus, to obtain a contradiction, we will show that if there is an obstruction set in $\Delta_k(q, D_{k+2})$ at i^{th} step, then there has to exist an obstruction set in $\Delta_k(q, D_{k+2})$ in $(i-1)^{th}$ step. The details of the proof are relegated to appendix. ◀

7 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. We showed that for any self-join-free or path query which is not coNP-hard, the algorithm correctly computes all certain answers. A similar fixpoint algorithm can be obtained for other kinds of constraints. It needs a few hypothesis such that being able to check in PTime whether a set of facts belongs to a repair. However, the analysis of this algorithm under other kinds of constraints is yet to be studied.

It is clear that when the fixpoint of the algorithm is bounded (*i.e.*, it converges after a number of steps which is independent of the input database) the certainty of the query can be expressed in first-order logic. It turns out that the converse is also true. Using the characterizations of [7] for self-join-free queries and of [6] for path queries, we can show that when the certainty can be expressed in first-order logic, then our algorithm is bounded. This will appear in the journal version of this paper.

As we have shown, our algorithm does not solve all the known cases where certainty can be solved in polynomial time. Hence, it would be interesting to have a (decidable)

636 characterization of the queries whose certainty can be solved using our algorithm; we leave
637 this for future work.

638 ## References

639 **1** Foto N. Afrati and Phokion G. Kolaitis. Repair checking in inconsistent databases: algorithms
640 and complexity. In Ronald Fagin, editor, *Database Theory - ICDT 2009, 12th International*
641 *Conference, St. Petersburg, Russia, March 23-25, 2009, Proceedings*, volume 361 of *ACM*
642 *International Conference Proceeding Series*, pages 31–41. ACM, 2009. doi:10.1145/1514894.
643 1514899.

644 **2** Marcelo Arenas, Leopoldo E. Bertossi, and Jan Chomicki. Consistent query answers in
645 inconsistent databases. In Victor Vianu and Christos H. Papadimitriou, editors, *Proceedings*
646 *of the Eighteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database*
647 *Systems, May 31 - June 2, 1999, Philadelphia, Pennsylvania, USA*, pages 68–79. ACM Press,
648 1999. doi:10.1145/303976.303983.

649 **3** Ariel Fuxman and Renée J. Miller. First-order query rewriting for inconsistent databases. *J.*
650 *Comput. Syst. Sci.*, 73(4):610–635, 2007. doi:10.1016/j.jcss.2006.10.013.

651 **4** John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ algorithm for maximum matchings in
652 bipartite graphs. *SIAM J. Comput.*, 2(4):225–231, 1973. doi:10.1137/0202019.

653 **5** Phokion G. Kolaitis and Enela Pema. A dichotomy in the complexity of consistent query
654 answering for queries with two atoms. *Inf. Process. Lett.*, 112(3):77–85, 2012. doi:10.1016/j.
655 ipl.2011.10.018.

656 **6** Paraschos Koutris, Xiating Ouyang, and Jef Wijsen. Consistent query answering for primary
657 keys on path queries. In Leonid Libkin, Reinhard Pichler, and Paolo Guagliardo, editors,
658 *PODS'21: Proceedings of the 40th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles*
659 *of Database Systems, Virtual Event, China, June 20-25, 2021*, pages 215–232. ACM, 2021.
660 doi:10.1145/3452021.3458334.

661 **7** Paraschos Koutris and Jef Wijsen. Consistent query answering for self-join-free conjunctive
662 queries under primary key constraints. *ACM Trans. Database Syst.*, 42(2):9:1–9:45, 2017.
663 doi:10.1145/3068334.

664 **8** Paraschos Koutris and Jef Wijsen. Consistent query answering for primary keys in datalog.
665 *Theory Comput. Syst.*, 65(1):122–178, 2021. doi:10.1007/s00224-020-09985-6.

666 **9** Jef Wijsen. A remark on the complexity of consistent conjunctive query answering under
667 primary key violations. *Inf. Process. Lett.*, 110(21):950–955, 2010. doi:10.1016/j.ipl.2010.
668 07.021.

A Details ommitted in Section 3

▶ **Example 4.** Consider again the query $q_2 : R_1(\underline{x}\ y) \wedge R_2(\underline{y}\ x)$ from Example 2. Let $k = 2$ and consider the execution of $\mathrm{Cert}_2(q_2)$. Initially, $\Delta_2(q_2, D)$ 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 $\Delta_2(q_2, D)$ (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. And so it continues.

We show that, for the query $q_2 : R_1(\underline{x}\ y) \wedge R_2(\underline{y}\ x)$ of Example 4, $\mathrm{Cert}_2(q_2)$ computes $\textsc{certain}(q_2)$. In other words, for q_2, $\mathrm{Cert}_2(q_2)$ is not an *under* approximation but an *exact* computation of certainty.

Observe that, for every repair r and fact α therein, there is at most one other fact α' in r such that $\{\alpha, \alpha'\} \models q_2$. 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 $\mathrm{Cert}_2(q_2)$ a complete procedure, as we shall see next.

In view of Proposition 3, it remains to show that if q_2 is certain for D then $\Delta_2(q_2, D)$ contains the empty set.

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 prove the following claim.

▷ Claim. If r is a minimal repair and $R_1(ab)$, $R_2(ba)$ are facts of r then $R_1(ab) \in \Delta_2(q_2, D)$.

Towards a contradiction, assume that $R_1(ab) \notin \Delta_2(q_2, D)$. We shall construct an infinite sequence $u_1, u_2, \ldots$ of distinct facts of D, contradicting the finiteness of D. We construct, at the same time, 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 invariant:

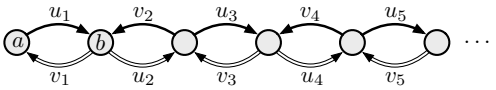
1. the u_i's are pairwise distinct;
2. $u_i \notin \Delta_2(q_2, D)$;
3. if $u_i = R_1(cd)$ then $v_i = R_2(dc)$ and if $u_i = R_2(cd)$ then $v_i = R_1(dc)$;
4. $u_{i+1} \sim v_i$ and $u_{i+1} \neq v_i$;
5. r_i is minimal and contain each u_j, $j \leq i$ together with v_i.

Initially $r_1 = r$, $u_1 = R_1(ab)$ and $v_1 = R_2(ba)$ and the invariant conditions are met, the second item being our initial hypothesis.

Consider step i. Consider the block B_i of v_i. As $u_i \notin \Delta_2(q_2, D)$ we know that B_i must contain an element u_{i+1} such that both $u_{i+1} \notin \Delta_2(q_2, D)$ and $\{u_i, u_{i+1}\} \notin \Delta_2(q_2, D)$. In particular $u_{i+1} \sim v_i$ but $u_{i+1} \neq v_i$ and items 2 and 4 of our inductive hypothesis are met. Towards the first item of our inductive hypothesis, if $u_{i+1} = u_j$ 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 $v_{i+1} \in r_{i+1}$. 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 $R_1(ab) \in \Delta_2(q_2, D)$ for any minimal repair containing both $R_1(ab)$ and $R_2(ba)$.

To conclude that $\mathrm{Cert}_2(q_2)$ returns the correct answer, consider a minimal repair r of D. As q_2 is certain for D, 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 $\Delta_2(q_2, D)$ and hence $\emptyset \in \Delta_2(q_2, D)$. 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 $R_1(ab') \in \Delta_2(q_2, D)$, as desired.

B Details ommitted in Section 4

▶ **Claim 11.** *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. ◀

▶ **Claim 12.** *Assume $\mathrm{PH}_\tau(i)$. 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 $\mathrm{PH}_\tau(i)$ 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 7 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 $\mathrm{PH}_\tau(i)$ 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 ◀

▶ **Theorem 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 $q \models \mathrm{PH}$.*

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 $q \models \mathrm{PH}_\tau$.

To prove $q \models \mathrm{PH}_\tau$, we need to show that $q \models \mathrm{PH}_\tau(i)$ 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 5 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 5 it follows that

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

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 or 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 $x \in vars(A^+)$. By ($\dagger$) this implies that $\mu(x) = \mu'(x)$ as desired. ◀

C Details ommitted in Section 5

▶ **Claim 20.** *For all $i \geq 0$, For $c \in adom(D)$ and for all prefix s_l of q if $\langle c, s_l \rangle \in N^i(q, D)$ then for all valid path Π where $last(\Pi) = c$ and $trace(\Pi) \sim s_l$ we have $S_\Pi \in \Delta^i(q, D)$.*

The proof of the Claim will make use of the following consequence of the fact that q is a factor of all words in $\mathcal{L}^{\curvearrowright}(q)$.

▶ **Lemma 25.** *For any path query q such that every word of $\mathcal{L}^{\curvearrowright}(q)$ contains q as factor, and prefix of q of the form $s_1 P R s_2 P R'$ for $s_1, s_2 \in \Sigma_q^*$ and $R \neq R'$, we have that $s_1 P$ is a suffix of $s_1 P R s_2 P$.*

Proof. Assume $q = s_1 P R s_2 P R' t$ and consider the word $w = s_1 P R s_2 P R s_2 P R' t$. A simple observation shows that $w \in \mathcal{L}^{\curvearrowright}(q)$. 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:

$$w(l) = w(l + |s_2| + 2) \tag{$\star$}$$

Let $w(i), w(i+1), \ldots, w(i+n-1)$ be the factor of w that matches q, and observe that $1 \leq i \leq |w| - |q| = |s_2| + 2$. If q is a suffix of w, it follows that $s_1 P$ is a suffix of $s_1 P R s_2 P$ and we are done. If q is not a suffix of w, then $i < |s_2| + 2$. Observe that $w((i-1) + |s_1 P R|) = R$

and $w((i-1)+|s_1PRs_2PR'|) = R'$. Hence, setting $l = (i-1)+|s_1PR| < |s_1|+|s_2|+3$, 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 20. The proof is by induction on i. For the base case $i = 0$, we have $N^0(q,D) = \{\langle c, q\rangle : c \in adom(D)\}$. Note that $\mathcal{A}^q$ has no ε transitions from a prefix of q to q. Hence, for any valid path $\Pi = u_1, \ldots, u_k$ such that $trace(\Pi) \sim q$ we must have $k = n$ and $trace(\Pi) = q$. In this case $(u_1, \ldots, u_n)$ forms a solution to q and hence $S_\Pi \in \Delta^0(q,D)$.

For the inductive step, let $\Pi = u_1, \ldots, u_k$ be any valid path such $trace(\Pi) \sim s_l$ and $last(u_k) = c$. Assuming $\langle c, s_l\rangle \in N^{i+1}(q,D)$, we will prove $S_\Pi \in \Delta^{i+1}(q,D)$. If $\langle c, s_l\rangle \in N^i(q,D)$ then by inductive hypothesis we have $S_\Pi \in \Delta^i(q,D)$ and we are done since $\Delta^i(q,D) \subseteq \Delta^{i+1}(q,D)$.

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 $R \in \Sigma_q$, and a fact $R(\underline{c}\ a)$ of D such that for every fact $R(\underline{c}\ b)$ of D we have $\langle b, s'\rangle \in N^i(q,D)$.

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

1. Case $s_k = q$. This case is similar to the base case. Since $(u_1, \ldots u_k)$ forms a solution to q, we have $S_\Pi \in \Delta^0(q,D) \subseteq \Delta^{i+1}(q,D)$.
2. Case $s_{k+1} = s_kR$.
   If there is already a fact of the form $u_i = R(\underline{c}\ b)$ in Π then the new path $\Pi' = u_1, u_2, \ldots, u_k, u_i$ is also a valid path where $last(\Pi') = b$. Also, since there is a run of $\mathcal{A}^q$ on $trace(\Pi)$ that ends at s_l, there is a run of $\mathcal{A}^q$ on $trace(\Pi')$ that ends at s'. Hence, $trace(\Pi') \sim s'$. Thus, by inductive hypothesis, $S_{\Pi'} \in \Delta^i(q,D)$. But since u_i is already present in Π, we obtain $S_\Pi = S_{\Pi'}$, and therefore $S_\Pi \in \Delta^i(q,D) \subseteq \Delta^{i+1}(q,D)$.
   Otherwise, for every fact u_i in Π if $trace(u_i) = R$ then $key(u_i) \neq c$. Take any arbitrary fact of the form $v = R(\underline{c}\ b)$ of D. The new path $\Pi' = u_1, u_2, \ldots, u_k, v$ is also valid and we have $last(\Pi') = b$. Again, since there is a run of the $\mathcal{A}^q$ on $trace(\Pi)$ that ends at s_l, there is also a run of $\mathcal{A}^q$ on $trace(\Pi')$ that ends at s'. Hence, $trace(\Pi') \sim s'$. Thus, by inductive hypothesis, we have $S_{\Pi'} = S_\Pi \cup \{R(\underline{c}\ b)\} \in \Delta^i(q,D)$. Since this holds for any fact of the form $R(\underline{c}\ b)$ we obtain, by definition of $\Delta(q,D)$, that $S_\Pi \in \Delta^{i+1}(q,D)$.
3. Case $s_{k+1} = s_kR'$ for some $R' \neq R$.
   Remember that $trace(\Pi) = s_k$ and there is a run of $\mathcal{A}^q$ on s_k that ends in state s_l. Let $s_l = s_1P$ (since s_l cannot be ε). Observe that, by definition of $\mathcal{A}^q$, a run on a word cannot end at a state of the form s_1P unless the word ends with P; therefore, s_k has to end with P. Altogether we have $s_{k+1} = \overbrace{\underbrace{s_1P}_{s_l} R\, s_2P}^{s_k} R'$ for some $s_1, s_2 \in \Sigma_q^*$ and $R \neq R'$. Applying Lemma 25, we obtain that $s_l = s_1P$ is a suffix of $s_k = s_1PRs_2P$. Let $\Pi' = u_{k-l+1}, \ldots, u_k$ be the suffix of Π such that $trace(\Pi') = s_l$. Note that Π' is a valid path and $S_{\Pi'} \subseteq S_\Pi$. Hence, it is sufficient to prove that $S_{\Pi'} \in \Delta^{i+1}(q,D)$. We are then in the situation of the already treated Case 2 above, since $s_{l+1} = s_lR$. Hence, $S_{\Pi'} \in \Delta^{i+1}(q,D)$. ◀

D Details ommitted in Section 6

Definition of D_n First we formally define the facts of the database D_n. Fix some $n \geq 4$. Recall that there are n blocks of the form $B_1, \cdots, B_n$ in D_n where each B_i consists of $n-1$ facts denoted $b_i^1, \cdots, b_i^{n-1}$ and there are $(n-1)(n-3)$ blocks of the form E_l^j for $1 \leq j \leq n-1, 1 \leq l \leq n-3$, where each E_l^j have two facts u_l^j and v_l^j.

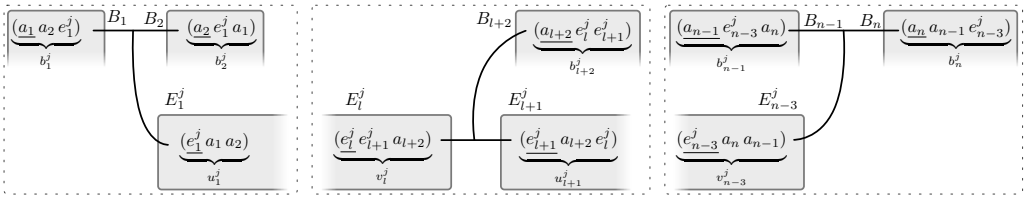


Figure 2 Triangles in the database D_n.

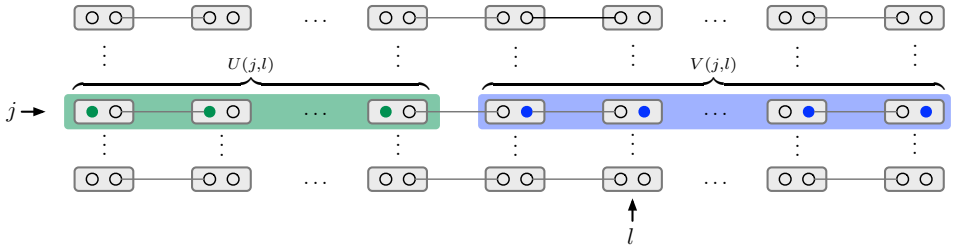


Figure 3 Depiction of $U(j,l)$ as green solid discs, and $V(j,l)$ as blue solid discs.

The database D_n has n active domain elements of the form $a_1, \cdots, a_n$ and $(n-1)(n-3)$ active domain element of the form e_i^j where $1 \leq j \leq n-1, 1 \leq i \leq n-3$. The facts of D_n are defined as follows.

- $b_1^j = (\underline{a_1} \;\; a_2 e_1^j)$, $b_2^j = (\underline{a_2} \;\; e_1^j a_1)$;
- for every $3 \leq i \leq n-2$: $\quad b_i^j = (\underline{a_i} \;\; e_{i-2}^j e_{i-1}^j)$;
- $b_{n-1}^j = (\underline{a_{n-1}} \;\; e_{n-3}^j a_n)$ and $b_n^j = (\underline{a_n} \;\; a_{n-1} e_{n-3}^j)$;
- $u_1^j = (\underline{e_1^j} \;\; a_1 a_2)$ and $v_{n-3}^j = (\underline{e_{n-3}^j} \;\; a_n a_{n-1})$;
- for every $1 \leq l < n-3$: $\quad v_l^j = (\underline{e_l^j} \;\; e_{l+1}^j a_{l+2})$ 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 $\{v_l^j, u_{l+1}^j, b_{l+2}^j\}$. By abstracting out the facts to black disks, we get the solution graph described in Figure 1.

▶ **Proposition 24.** *Let $k \geq 2$. $D_{k+2} \not\models \mathrm{Cert}_k(q)$.*

It remains to prove Proposition 24. For this, we first setup up some definitions and prove some useful lemmas. When D_n is clear from the context, we denote $\mathbb{B} = \{B_1, \ldots B_n\}$ and, for every $1 \leq j \leq n-1$, we denote $\mathbb{E}^j = \{E_l^j \mid 1 \leq l \leq n-3\}$ 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 $l \leq k$ 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 $W[\mathbb{E}^j]$ 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 $U(j,l) = \{u_k^j \mid 1 \leq k \leq l-2\}$ and $V(j,l) = \{v_k^j \mid l-1 \leq k \leq n-3\}$, 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 $u^j_{l-1},\ v^j_{l-2} \notin U(j,l) \cup V(j,l)$. Consequently, there are no solutions in the set of facts $U(j,l) \cup V(j,l) \cup \{b^j_l\}$.

Similarly, recall that for every $1 \leq j \leq n-1$, D_n contains the triangle $\{b^j_1, b^j_2, u^j_1\}$, so if a repair contains either b^j_1 of b^j_2 and contains no solution to q, this repair must pick no u fact in the blocks of $\mathbb{E}^j$. Thus, we define $U(j,1) = U(j,2) = \emptyset$ and $V(j,1) = V(j,2) = \{v^j_k \mid 1 \leq k \leq n-3\}$, so that there are no solutions in the set of facts $U(j,l) \cup V(j,l) \cup \{b^j_l\}$, for $l \in \{1,2\}$.

Dually, $\{b^j_{n-1}, b^j_n, v^j_{n-3}\}$ form a triangle, so define $U(j,n) = U(j,n-1) = \{u^j_k \mid 1 \leq k \leq n-3\}$ and $V(j,n) = V(j,n-1) = \emptyset$. Again there are no solutions in the set of facts $U(j,l) \cup V(j,l) \cup \{b^j_l\}$, for $l \in \{n-1, n\}$.

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

$$U(j,l) \cup V(j,l) \cup \{b^j_l\} \text{ forms a partial repair over the blocks } \{B_l\} \cup \mathbb{E}^j \text{ such that there is no solution within this partial repair.} \tag{h6}$$

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

1. $j_1, j_2, \dots j_l$ are pairwise distinct.
2. For every $1 \leq m \leq l$ we have $W[\mathbb{E}^{j_m}] \subseteq U(j_m, i_m) \cup V(j_m, i_m)$.
3. For every $1 \leq j \leq n-1$ there exists $1 \leq l \leq n$ such that $W[\mathbb{E}^j] \subseteq U(j,l) \cup V(j,l)$.

▶ **Lemma 26.** *If W is a k-obstruction set, then there are no solutions to the query q within W.*

Proof. Remember that W is 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 $b^j_l \in W[\mathbb{B}]$. By construction of D_n this solution must involve either u^j_{l-1} or v^j_{l-2}. But by Condition (2) none of them are in W. ◀

To prove Proposition 24, we shall make use of the following two claims:

1. 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.
2. For every set of facts W, if W is a k-obstruction set, then $W \notin \Delta_k(q, D_{k+2})$.

The two claims 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 $W \notin \Delta_k(q, D_{k+2})$. Hence, $D_{k+2} \not\models \mathrm{Cert}_k(q)$.

Proof of Claim 1. 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 $E^j_1 \dots E^j_{k-1}$.

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 set W to $\{b^1_{i_1}, b^2_{i_2}, \dots b^m_{i_m}\}$. Moreover for every $1 \leq j \leq m$ we add to W the partial repair induced by $U(j,i_j) \cup V(j,i_j)$. That is, we add $(U(j,i_j) \cup V(j,i_j)) \cap \bigcup \mathbb{X}$ to W (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. ◀

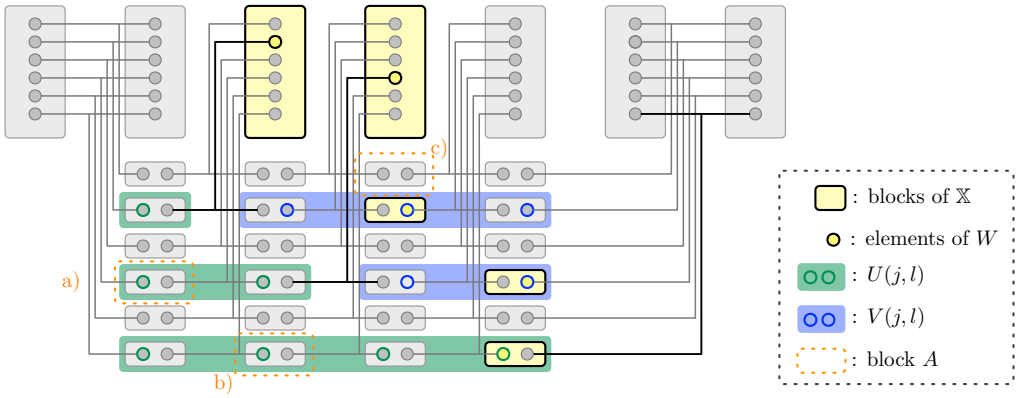


Figure 4 Database D_{k+2} in the proof of Claim 2, with obstructing set W over blocs $\mathbb{X}$. Dotted-line boxes depict the block A in cases a) b) and c) of the proof.

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

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

Note that $n = 0$ is not possible since W does not contain any solution by Lemma 26; hence $n > 0$. By definition of $\Delta_k^n(q, D_{k+2})$, this implies that there exists a block A of D_{k+2} such that $A \notin \mathbb{X}$ and for all $a \in A$ there is a subset $W' \subseteq W \cup \{a\}$ of size at most k such that $W' \in \Delta_k^{n-1}(q, D_{k+2})$ and further W' is not a k-obstruction set, nor a subset thereof.

Let $\mathbb{X} \cap \mathbb{B} = \{B_{i_1}, \ldots B_{i_m}\}$ for some $m \leq k$ and $W[\mathbb{B}] = \{b_{i_1}^{j_1}, b_{i_2}^{j_2}, \ldots b_{i_m}^{j_m}\}$. So there exists $k+1-m$ many distinct indices j that are not in $\{j_1, \ldots, j_m\}$. Let $j_{m+1}, \ldots, j_k, j_{k+1}$ 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 $s \notin \{j_1, \ldots j_m\}$ 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-sized subset of $W \cup \{b_l^s\}$ forms a k-obstruction set since condition (2) holds vacuously for s. Hence, there must exist a k-obstruction set inside $\Delta_k^{n-1}(q, D_{k+2})$, which is in contradiction with the minimality of n.
- If, otherwise, $A = E_l^r$ for some $1 \leq r \leq k+1$ 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 $r \in \{j_1, \ldots j_m\}$, say $r = j_t$, then by condition (2) $W[\mathbb{E}^r] \subseteq U(j_t, i_t) \cup V(j_t, i_t)$. Let w be the fact in E_l^r 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 $r \notin \{j_1, \ldots j_m\}$ but $\mathbb{X} \cap \mathbb{E}^r \neq \emptyset$ then by condition (3), $W[\mathbb{E}^r] \subseteq U(r, r') \cup V(r, r')$ for some $1 \leq r' \leq k+2$. Let w be the fact in E_l^r picked by $U(r, r') \cup V(r, r')$. 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, $r \notin \{j_1, \ldots j_m\}$ and $\mathbb{X} \cap \mathbb{E}^r = \emptyset$. In this case, pick $v_l^r \in E_l^r$ for some arbitrary l. We have $\{v_l^r\} \subseteq U(r, 1) \cup V(r, 1)$. Hence, it can be verified that every k-sized subset of $W \cup \{v_l^r\}$ is a k-obstruction set, which again is a contradiction. ◀

▶ **Theorem 27.** *Let $q = R(\underline{x}\ yz) \wedge R(\underline{z}\ xy)$. Then* CERTAIN$(q)$ *is complete for bipartite matching under* FO/ LogSpace *reductions.*

Proof. From Lemma 22 it follows that we can reduce CERTAIN(q) to bipartite matching. The reader can verify that the reduction is FO/ LogSpace. We now prove the other direction.

Given a bipartite graph $G = (V_1 \cup V_2, E)$, let $V_1 = \{s_1, \ldots s_n\}$ and $V_2 = \{t_1, \ldots t_m\}$. Consider the problem of determining whether there exists a matching that saturates V_1. We will reduce this problem to CERTAIN(q).

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 matching that saturates V_1. So assume that for every $s_j \in V_1$, $N(s_j) \neq \emptyset$. 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. So we can also assume that for every $t_i \in V_2$, $N(t_i) \neq \emptyset$. Further, suppose there is some t_i such that $|N(t_i)| = 1$, let s_j be the single neighbour of t_i. In this case, if there exists a matching that saturates V_1 then there is a matching that saturates V_1 where s_j is matched with t_i. So we can remove the pair (s_j, t_i) from the input. This means that we can assume that for every $t_i \in V_2$, $|N(t_i)| \geq 2$.

Altogether, we have $|N(s_j)| \geq 1$ for all $s_j \in V_1$ and $|N(t_i)| \geq 2$ for all $t_i \in V_2$. Note that these properties can be checked in FO/ LogSpace, linear time.

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 21.

- 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 $s_{i_1}, \ldots s_{i_l} \in V_1$ be the neighbours of t. By 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. 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 (the three facts form a triangle).

  If $l \geq 4$ then create $l - 3$ new blocks denoted by $E^i_1, \ldots E^i_{l-3}$ 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, 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 FO/ LogSpace. For each such j and l we again define $U(i, l) = \{u^i_k \mid 1 \leq k \leq l - 2\}$ and $V(i, l) = \{v^i_k \mid l - 1 \leq k \leq l - 3\}$,

Now suppose there is a matching that saturates V_1 then 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 $E^i_1 \ldots E^i_l$.

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 $E^i_1 \ldots E^i_l$. It can be verified that the obtained repair does not contain any solution.

Finally, suppose there is a repair over G_D that falsifies the query then 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 the repair since that will make the query 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 matching that maps every $s_j \in V_i$ to $t_i \in V_2$ where b^i_j is the fact in the falsifying repair from the block B_j. ◀