# A Dichotomy in the Complexity of Consistent Query Answering for Two Atom Queries With Self-Join

Anantha Padmanabha, IIT Dharwad
Luc Segoufin, INRIA, ENS-Paris
Cristina Sirangelo, Université Paris Cité, CNRS, Inria, IRIF

April 30, 2024

**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 show that it also holds for queries with two atoms.

## 1 Introduction

A relational database often comes with integrity constraints. With the attempts to harness data from complex environments like big data, social media etc., where the database is built by programs that go over a large data dump, more often than not we end up with a database that violates one or more of the integrity constraints. This is because in such heterogeneous sources, the data is often incomplete or ambiguous. Inconsistencies in databases also occur while integrating data from multiple sources.

To deal with this problem, one approach is to *clean* the database when it is being built and/or modified. However, this task is not easy as it is inherently non-deterministic: there may be many equally good candidate tuples to add/delete/update to make the database consistent. In the absence of additional information (which is often the case), these decisions are arbitrary.

There is another way to cope with this problem: we allow the database to be inconsistent and the problem is handled during query evaluation. In this approach, we retain the inconsistent database as it is and we rely on the notion of database *repair*. Intuitively repairing a database corresponds to obtaining a consistent database by making minimal changes to the inconsistent one. A conservative approach to evaluate a query is to evaluate it over every possible repair and retain only the *certain answers*, *i.e.* the query answers which are true on all repairs. This approach is called *consistent query answering* [ABC99, Ber19].

This approach for handling inconsistency has an advantage of not loosing any information and avoids making arbitrary choices to make the database consistent. However, since we need to evaluate the query on all the repairs, this will affect the complexity of query evaluation. The impact will of course depend on the type of integrity constraints and on the definition of a repair; but most often there could be exponentially many ways to repair a database.

For a fixed boolean conjunctive query $q$, the decision version of the certain query answering problem is the following: given an inconsistent database $D$ as input, does $q$ evaluate to true on all the repairs of $D$?

When we consider primary key constraints, the natural notion of a repair is to pick exactly one tuple for every primary key. Thus, every repair is a subset of the given inconsistent database. But there could be exponentially many repairs for the given database. For a fixed boolean conjunctive query, in the presence of primary key constraints, checking for certain answers is in coNP. This

is because, to check that the query is not certainly true, it is enough to guess a subset of the database which forms a repair and verify that it makes the query false. However, there are queries for which the problem can be solved in PTIME and for some queries, the problem is coNP-hard.

The main conjecture for consistent query answering in the presence of primary keys is that there are no intermediate cases: for a fixed boolean conjunctive query $q$, the consistent query answering problem for $q$ is either solvable in PTIME or coNP-complete.

The conjecture has been proved for self-join-free Boolean conjunctive queries [KW17] and path queries [KOW21]. However, the conjecture remains open for arbitrary conjunctive queries, in particular for queries having self-joins (*i.e.* having at least two different atoms using the same relation symbol).

In this paper we show that the conjecture holds for conjunctive queries with two atoms. As the case of self-join-free queries has already been solved [KP12], we consider only queries consisting of two atoms over the same relation symbol.

Towards proving the conjecture we start by introducing the notion of 2way-determinacy and we distinguish two separate cases. The first case proves the dichotomy for all the two-atom queries with self-joins that are not 2way-determined; these are identified via syntactic conditions. For these queries coNP-hardness is obtained through a reduction from the self-join-free case with two atoms [KP12]. On the other hand tractable cases are obtained via the greedy fixpoint algorithm developed in [FPSS23] for self-join free queries.

For queries that are 2way-determined we use a semantic characterization. To this end we introduce the notion of TRIPATH, which is a database of a special form, and further classify TRIPATH into triangle-TRIPATH and fork-TRIPATH. For 2way-determined queries the existence of a fork-TRIPATH establishes the desired complexity dichotomy. In particular we prove that the certain answering problem is coNP-hard for queries which admit a fork-TRIPATH while it is in PTIME otherwise. In the latter case the polynomial time algorithm is a combination of the greedy fixpoint algorithm of [FPSS23] and a bipartite matching-based algorithm (also introduced in [FPSS23]).

Our second result further refines the polynomial time case by identifying classes of queries for which the greedy fixpoint algorithm correctly computes certain answers (assuming PTIME $\neq$ coNP). The frontier turns out to be the presence of a triangle-TRIPATH. Indeed we show that for 2way-determined queries which do not admit a TRIPATH at all (neither fork-TRIPATH, nor triangle-TRIPATH) the greedy fixpoint algorithm alone is correct. Furthermore we also prove that this algorithm fails to compute the certain answer to 2way-determined queries admitting a triangle-TRIPATH.

**Related work** The case of *self-join-free* conjunctive queries with two atoms was considered by Kolaitis and Pema [KP12]. Proving the dichotomy in the presence of self-joins requires a completely different technique. However we use the coNP-complete characterization of [KP12] for solving a special case in our analysis.

We rely heavily on the polynomial time algorithms developed in [FPSS23]. In this work a simple greedy fixpoint algorithm, referred to as $\mathrm{Cert}_k(q)$, was introduced and was shown to solve all the PTIME cases of self-join-free conjunctive queries (and also path queries). Moreover [FPSS23] shows that some two-atom queries with self-join which are in PTIME cannot be solved by $\mathrm{Cert}_k(q)$, but a different algorithm based on bipartite matching will work for them. We essentially show that a combination of these two algorithms solve all the polynomial time cases of two-atom conjunctive queries. In essence we show that if the combination of the two algorithms does not work then the query is coNP-hard.

We do not rely on the notion of attack graph or any other tools used for self-join queries, developed by Koutris and Wijsen [KW17].

## 2 Preliminaries

We consider boolean conjunctive queries over relational databases. As our queries will have only two atoms and because the self-join-free case is already solved, we can assume that these two

atoms refer to the same relational symbol. Therefore we consider relational schema with only one relational symbol, associated with a primary key constraint.

A relational schema consist of a relation symbol $R$ with signature $[k, l]$, where $k \geq 1$ denotes the arity of $R$ and the first $l$ ($\geq 0$) positions form the primary key of $R$.

We assume an infinite domain of **elements** and an infinite set of variables. A **term** is of the form $R(\bar{t})$ where $\bar{t}$ is a tuple of elements or variables of arity $k$. A term $R(\bar{t})$ is called a **fact** if $\bar{t}$ is a tuple of elements, and $R(\bar{t})$ is called an **atom** if $\bar{t}$ is a tuple of variables. We use $a, b, c$ etc to denote facts and $A, B, C$ etc to denote atoms.

Given a term $R(\bar{t})$ we let $R(\bar{t})[i]$ denote the variable / element at $i$-th position of $\bar{t}$. For a set of positions $I$ we let $R(\bar{t})[I] = \{R(\bar{t})[i] \mid i \in I\}$. Let $S$ be the set of all $k$ positions of $R$. If $a$ is a fact then we write $adom(a)$ for $a[S]$. Similarly if $A$ is an atom then we write $vars(A)$ for $A[S]$. We define the **key** of $R(\bar{t})$ to be the tuple $\overline{key}(R(\bar{t}))$ consisting of the first $l$ elements of $\bar{t}$ and let $key(R(\bar{t})) = R(\bar{t})[K]$, where $K$ is the set of the first $l$ (key) positions of $R$. For instance, if $R$ has signature $[5, 3]$ and $A = R(xyx\ yz)$, we have $\overline{key}(A) = (x, y, x)$, $key(A) = \{x, y\}$ and $vars(A) = \{x, y, z\}$. Two terms $R(\bar{t}_1)$ and $R(\bar{t}_2)$ are key-equal if $\overline{key}(R(\bar{t}_1)) = \overline{key}(R(\bar{t}_2))$ and we denote it by $R(\bar{t}_1) \sim R(\bar{t}_2)$.

A **database** is a *finite* set of facts. We say that a database $D$ is of size $n$ if there are $n$ facts in $D$. A database $D$ is **consistent** if it does not contain two distinct key-equal facts. A **block** in $D$ is a maximal subset of $D$ that contains key-equal facts. A **repair** of $D$ is a $\subseteq$-maximal consistent subset of $D$. Note that $D$ can be partitioned into disjoint blocks and every repair picks exactly one fact from every block. If $r \subseteq D$ is a repair and $a$ is a fact in $D$ then for any $a' \sim a$ we denote as $r[a \to a']$ the repair obtained from $r$ by replacing $a$ by $a'$.

A **query** $q$ is given by two atoms $A$ and $B$ and it corresponds to the Boolean conjunctive query $\exists \bar{y}\ A \wedge B$ where $\bar{y}$ is the tuple of all the variables in $vars(A) \cup vars(B)$. Since every variable is quantified, we ignore the quantification and write $q = AB$. For instance, if the query is $q = \exists xyzu\ R(xyx\ uz) \wedge R(yxu\ zu)$ then we let $A = R(xyx\ uz)$, $B = R(yxu\ zu)$ and write $q = AB$.

A database $D$ satisfies a query $q = AB$, denoted by $D \models q$ (sometimes denoted by $D \models AB$), if there exists a mapping $\mu$ from $vars(A) \cup vars(B)$ to elements such that $\mu(A), \mu(B) \in D$. In this case the pair $(\mu(A), \mu(B))$ of (not necessarily distinct) facts of $D$ is called a **solution** to $q$ in $D$. We also say that the fact $\mu(A)$ matches $A$ and $\mu(B)$ matches $B$. Different mappings give different solutions. The set of solutions to $q$ in $D$ is denoted by $q(D)$. We will also write $D \models q(ab)$ to denote that $(a, b)$ is a solution to $q$ in $D$ via some $\mu$. We also write $D \models q\{ab\}$ to denote $D \models q(ab)$ or $D \models q(ba)$. If $D$ is clear from the context we simply write $q\{ab\}$, $q(ba)$ etc.

A query $q$ is **certain** for a database $D$ if all repairs of $D$ satisfy $q$. For a fixed query[1] $q$, we denote by **certain**$(q)$ the problem of determining, given a database $D$, whether $q$ is certain for $D$. We write $D \models \textsc{certain}(q)$ or $D \in \textsc{certain}(q)$ to denote that $q$ is certain for $D$. Clearly the problem is in coNP as one can guess a (polynomial sized) repair $r$ of $D$ and verify that $r$ does not satisfy $q$.

We aim at proving the following result:

**Theorem 1.** *For every (2-atom) query $q$, the problem* $\textsc{certain}(q)$ *is either in* PTime *or* coNP*-complete.*

Note that $\textsc{certain}(q)$ is trivial if $q$ has only one atom. As we deal with data complexity, it is then also trivial for any query equivalent (*over all consistent databases*) to a query with one atom. For a query $q = AB$ this can happen in two cases: (1) there is a homomorphism from $A$ to $B$ or from $B$ to $A$ ; (2) $\overline{key}(A) = \overline{key}(B)$ (the query is then always equivalent over consistent databases, to a single atom $R(C)$ where $C$ is the most general term that has homomorphism from both $A$ and $B$.) Hence, we will assume in the rest of this paper that $q = AB$ is such that $\overline{key}(A) \neq \overline{key}(B)$ and $q$ is not equivalent to any of its atoms.

In the rest of the paper we will often underline the first $l$ positions of an atom or a fact in order to highlight the primary-key positions. We then write $R(\underline{xyz}\ uv)$ to denote an atom involving a relation of signature $[5, 3]$. Similarly we write $R(\underline{\alpha\beta\gamma}\ \delta\varepsilon)$ to denote a fact over the same signature.

[1] Along standard lines, we adopt the *data complexity* point of view, *i.e.* the query is fixed and we measure the complexity as a function on the number $n$ of facts in $D$.

# 3 Dichotomy classification

The decision procedure for deciding whether the certainty of a query is hard or easy to compute works as follows.

• We first associate to any query a canonical self-join-free query by simply renaming the two relation symbols. If certainty of the resulting query is hard, and this can be tested using the syntactic characterization of [KP12], then it is also hard for the initial query. This is shown in Section 4.

• In Section 6 we give a simple syntactic condition guaranteeing that the greedy polynomial time fixpoint algorithm of [FPSS23] (presented in Section 5) computes certainty.

The remaining queries, where the two syntactic tests mentioned above fail, are called 2way-determined. They enjoy some nice semantic properties that are described in Section 7 and that we exploit to pinpoint their complexity. Towards this, we define in Section 7 a special kind of database called TRIPATH whose solutions to the query have a particular structure. We distinguish two variants of TRIPATH, as fork-TRIPATH and triangle-TRIPATH.

• If the query does not admit any TRIPATH (*i.e.* neither fork TRIPATH nor triangle TRIPATH) then certainty can be computed using the greedy fixpoint algorithm as show in Section 8.

• If the query admits a fork-TRIPATH, then certainty is coNP-hard as shown in Section 9.

• Finally, for queries that admits a triangle-TRIPATH but no fork-TRIPATH, certainty can be computed in PTIME. This is shown in Section 10. For such queries, we prove that the algorithm of Section 5 is not expressive enough to compute certainty. However we prove that a combination of it together with a known bipartite matching-based algorithm (again from [FPSS23]) is correct.

# 4 First coNP-hard case

Given a query $q = AB$, we can associate it with a canonical self-join-free query *sjf*($q$) over a schema with two distinct relational symbols[2] $R_1$ and $R_2$ of the same arity as $R$. The query *sjf*($q$) is defined by replacing $R$ by $R_1$ in $A$ and $R$ by $R_2$ in $B$. For instance, if $q_1 = R(\underline{xu}\ xv) \wedge R(\underline{vy}\ uy)$ then *sjf*($q_1$) $= R_1(\underline{xu}\ xv) \wedge R_2(\underline{vy}\ uy)$. Intuitively, *sjf*($q$) is the same query as $q$ but with two different relation names.

We show that computing the certainty of $q$ is always harder than computing the certainty of *sjf*($q$). This is the only place where we use the assumption that $q$ is not equivalent to a one-atom query.

**Proposition 2.** *Let $q$ be a query. There is a polynomial time reduction from* CERTAIN(*sjf*($q$)) *to* CERTAIN($q$).

*Proof sketch.* Assume $q = AB$ where $A$ and $B$ are atoms using the relational symbol $R$. Let $R_1$ and $R_2$ be the symbols used in *sjf*($q$). Let $D$ be a database containing $R_1$-facts and $R_2$-facts. We construct in polynomial time a database $D'$ containing $R$-facts such that $D \models$ CERTAIN(*sjf*($q$)) iff $D' \models$ CERTAIN($q$).

For every fact $a = R_1(\bar{u})$ of $D$, let $\mu(a) = R(\bar{v})$ be a fact where every position $i$ of $\bar{v}$ is the pair $\langle z, \alpha \rangle$ where $z$ is the variable at position $i$ in $A$ while $\alpha$ is the element at position $i$ in $\bar{u}$. Similarly if $a = R_2(\bar{u})$ then $\mu(a) = R(\bar{v})$ where every position $i$ of $\bar{v}$ is the pair $\langle z, \alpha \rangle$ where $z$ is the variable at position $i$ in $B$ while $\alpha$ is the element at position $i$ in $\bar{u}$. Let $D' = \mu(D)$. It turns out that $D'$ has the desired property and this requires that $q$ is not equivalent to a one-atom query. □

It follows from Proposition 2 that whenever *sjf*($q$) is coNP-hard then CERTAIN($q$) is also coNP-hard. It turns out that we know from [KP12] which self-join-free queries with two atoms are hard. This yields the following result.

**Theorem 3.** *Let $q = AB$ be such that both the following conditions hold :*

[2] In section 2 we have defined all the notions with respect to a single relation in the vocabulary. In this section, and only here, we consider two relations. Since the definitions are standard, we will not state them explicitly.

1. *vars*$(A) \cap$ *vars*$(B) \not\subseteq$ *key*$(A)$ *and* *vars*$(A) \cap$ *vars*$(B) \not\subseteq$ *key*$(B)$ *and* *key*$(A) \not\subseteq$ *key*$(B)$ *and* *key*$(B) \not\subseteq$ *key*$(A)$*;*
2. *key*$(A) \not\subseteq$ *vars*$(B)$ *or* *key*$(B) \not\subseteq$ *vars*$(A)$.

*Then* CERTAIN$(q)$ *is* coNP*-complete.*

For instance we can deduce from Theorem 3 that the query $q_1$ mentioned above is such that CERTAIN$(q_1)$ is coNP-complete since $u$ and $v$ are shared variables but $u \notin key(B)$, $v \notin key(A)$, moreover $key(B) \not\subseteq key(A)$ and $x \in key(A)$ but is not in $vars(B)$.

Note that the converse of Proposition 2 is not true. For instance, the query $q_2 = R(\underline{xu}\ xy) \wedge R(\underline{uy}\ xz)$ is such that CERTAIN$(sjf(q_2))$ can be solved in polynomial time by the characterization of [KP12], but as we will see, CERTAIN$(q_2)$ is coNP-hard.

# 5 The greedy fixpoint algorithm

In most of the cases where we prove that CERTAIN$(q)$ is in PTIME, we use the following greedy fixpoint algorithm which was introduced in [FPSS23]. For a fixed query $q$ and $k \geq 1$, we define the algorithm $\text{Cert}_k(q)$. It takes a database $D$ as input and runs in time $O(n^k)$ where $n$ is the size of $D$. For a database $D$, a set $S$ of facts of $D$ is called a $k$-set if $|S| \leq k$ and $S$ can be extended to a repair (*i.e.* S contains at most one fact from every block of $D$).

The algorithm inductively computes a set $\Delta_k(q, D)$ of $k$-sets while maintaining the invariant that for every repair $r$ of $D$ and every $S \in \Delta_k(q, D)$ if $S \subseteq r$ then $r \models q$. The algorithm returns *yes* if eventually $\emptyset \in \Delta_k(q, D)$. Since all repairs contain the empty set, from the invariant that is maintained, it follows that $D \models$ CERTAIN$(q)$. The set $\Delta_k(q, D)$ is computed as follows:

Initially $\Delta_k(q, D)$ contains all $k$-set $S$ such that $S \models q$. Clearly, this satisfies the invariant. 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 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.

For a fixed $k$, we write $D \models \mathbf{Cert}_k(q)$ or $D \in \mathbf{Cert}_k(q)$ to denote that $\text{Cert}_k(q)$ returns *yes* upon input $D$. Note that $\text{Cert}_k(q)$ is always an under-approximation of CERTAIN$(q)$, *i.e.* whenever $\text{Cert}_k(q)$ returns *yes* then $q$ is certain for the input database. However, $\text{Cert}_k(q)$ could give false negative answers. In [FPSS23] it is proved that this algorithm captures all polynomial time cases for self-join-free queries and path queries by choosing $k$ to be the number of atoms in the query.

# 6 First polynomial time case

In view of Theorem 3, it remains to consider the case where one of the conditions of Theorem 3 is false. In this section we prove that if the condition (1) is false for $q$ then CERTAIN$(q)$ is in PTIME. By symmetry we only consider the case where *vars*$(A) \cap$ *vars*$(B) \subseteq$ *key*$(B)$ or *key*$(A) \not\subseteq$ *key*$(B)$. The other case follows from the fact that $q = AB$ is equivalent to the query $BA$.

**Theorem 4.** *Let* $q = AB$ *be such that* *key*$(A) \subseteq$ *key*$(B)$ *or* *vars*$(A) \cap$ *vars*$(B) \subseteq$ *key*$(B)$*. Then* CERTAIN$(q) = \text{Cert}_2(q)$*, hence* CERTAIN$(q)$ *is in* PTIME*.*

From Theorem 4 it follows that the complexity of computing certain answers for queries like $q_3 = R(\underline{x}\ y) \wedge R(\underline{y}\ z)$ and $q_4 = R(\underline{xx}\ uv) \wedge R(\underline{xy}\ ux)$ is in PTIME. In the case of $q_3$ this is because the only shared variable is $y$ and $key(\overline{B}) = \{y\}$. In the case of $q_4$ this is because $key(A) = \{x\} \subseteq \{xy\} = key(B)$.

In the remaining part of this section we prove Theorem 4. The main consequence of the assumption on the query is the following zig-zag property. We say that $q$ satisfies the zig-zag property if for all database $D$, for all facts $a, b, b', c$ of $D$ such that $a \not\sim c$, $a \neq b$ and $b \sim b'$, if $D \models q(ab)$ and $D \models q(cb')$ then $D \models q(ab')$.

**Lemma 5.** *Let $q = AB$ be such that $vars(A) \cap vars(B) \subseteq key(B)$ or $key(A) \subseteq key(B)$. Then $q$ satisfies the zig-zag property.*

The key to the proof of Theorem 4 is the following lemma.

**Lemma 6.** *Let $q = AB$ be a query satisfying the zig-zag property. For all databases $D$ and for all repair $r$ of $D$ if $r \models q(ab)$ then $\{a\} \in \Delta_2(q, D)$ or there exists a repair $s$ of $D$ such that $q(s) \subsetneq q(r)$.*

*Proof sketch.* Assume that $\{a\} \notin \Delta_2(q, D)$. This means that there exists some $b' \sim b$ such that $\{a, b'\} \notin \Delta_2(q, D)$. Then consider $r' = r[b \to b']$. Notice that the only new solutions in $r'$ that are not in $r$ should involve $b'$. Hence if $b'$ is not a part of any solution, then $r'$ is the required repair. Otherwise note that $b'$ can only be a part of solutions of the form $r' \models q(b'c)$ (if $r' \models q(cb')$ for some $c$ then by zig-zag property we we also have $r' \models q(ab')$ which is a contradiction). We also have $\{b'\} \notin \Delta_2(q, D)$ (otherwise $\{a, b'\} \in \Delta_2(q, D)$ which is a contradiction) and we can repeat the construction. This way, we inductively build a sequence of repairs $r_0, r_1, \ldots r_n$ such that $r_0 = r$ and $r_n$ is the desired repair. □

*Proof of Theorem 4.* Let $D \models \text{CERTAIN}(q)$. We prove that $D \models \text{Cert}_2(q)$.

We first show that every repair $r$ of $D$ contains some fact $a \in r$ such that $\{a\} \in \Delta_2(q, D)$. Pick an arbitrary repair $r$ of $D$.

Let $r'$ be a minimal repair having $q(r') \subseteq q(r)$ (possibly $r' = r$). Since all the repairs contain a solution, there exist facts $a, b$ of $r'$ such that $r' \models q(ab)$. By Lemma 5, $q$ satisfies the zig-zag property, thus we can apply Lemma 6. This implies that $\{a\} \in \Delta_2(q, D)$, otherwise one can construct another repair $s$ of $D$ such that $q(s) \subsetneq q(r')$, contradicting minimality of $r'$. By the choice of $r'$, one has also $r \models q(ab)$, thus we have $a \in r$ such that $\{a\} \in \Delta_2(q, D)$.

Now let $r_{min}$ be a repair of $D$ containing the minimum number of facts $a$ such that $\{a\} \in \Delta_2(q, D)$. Let $m$ be this minimum number. By the property proved above there exists a fact $b \in r_{min}$ such that $\{b\} \in \Delta_2(q, D)$. We claim that for all $b' \sim b$, we have $\{b'\} \in \Delta_2(q, D)$. Suppose not, then $r_{min}[b \to b']$ contains $m - 1$ facts in $\Delta_2(q, D)$, contradicting minimality of $r_{min}$.

Overall this proves $\emptyset \in \Delta_2(q, D)$ and hence $D \models \text{Cert}_2(q)$. □

# 7 2way-determined queries

From Theorem 3 and Theorem 4 it remains to consider the case where condition (1) of Theorem 3 is true and condition (2) is false. Thus we can assume that the query satisfies the following conditions[3]:

$$key(A) \not\subseteq key(B) \text{ and } key(B) \not\subseteq key(A) \text{ and}$$
$$key(A) \subseteq vars(B) \text{ and } key(B) \subseteq vars(A)$$

We call such queries **2way-determined**. Queries that are 2way-determined have special properties that we will exploit to pinpoint the complexity of their consistent evaluation problem. They are summarized in the following lemma.

**Lemma 7.** *Let $q$ be a 2way-determined query. Then for all database $D$ and for all facts $a, b, c \in D$ suppose $D \models q(ab)$ then :*

- *if $D \models q(ac)$ then $c \sim b$*
- *if $D \models q(cb)$ then $c \sim a$*

*Proof.* Assume $q = AB$ and $D \models q(ac)$. As $key(B) \subseteq vars(A)$ it follows that that $\overline{key}(c) = \overline{key}(b)$. The second claim is argued symmetrically using $key(A) \subseteq vars(B)$. □

[3]We can drop the condition $vars(A) \cap vars(B) \not\subseteq key(A)$ because $key(A) \not\subseteq key(B)$ and $key(B) \subseteq vars(A)$ together imply $vars(A) \cap vars(B) \not\subseteq key(A)$ (and we drop $vars(A) \cap vars(B) \not\subseteq key(B)$ symmetrically).

In other words, within a repair, a fact can be part of at most two solutions. Moreover, when a fact $e$ is part of two solutions of the repair, the solutions must be of the form $q(de)$ and $q(ef)$. We then say that the fact $e$ is **branching** (with $d$ and $f$). If in addition $q(fd)$ holds then we say that $def$ is a **triangle**, otherwise $def$ is a **fork**. The facts that can potentially be part of two solutions in a repair play a crucial role in our proofs.

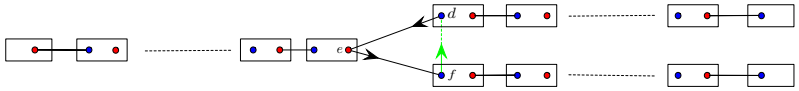
When $q$ is 2way-determined, the complexity of CERTAIN($q$) will depend on the existence of a database, called TRIPATH, whose solutions to $q$ can be arranged into a tree-like shape with one branching fact as specified next.

Let $d, e, f$ be three facts of a database $D$ such that $e$ is branching with $d, f$. Depending on the key inclusion conditions of $def$, we define $\bar{g}(e)$ as follows:

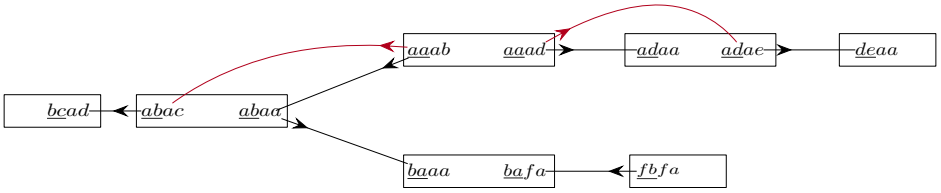
| | | |
|---|---|---|
| if $key(d) \subseteq key(e)$ and $key(f) \not\subseteq key(e)$ | then | $\bar{g}(e) = \overline{key}(d)$ |
| if $key(d) \not\subseteq key(e)$ and $key(f) \subseteq key(e)$ | then | $\bar{g}(e) = \overline{key}(f)$ |
| if $key(d) \subseteq key(f) \subseteq key(e)$ | then | $\bar{g}(e) = \overline{key}(d)$ |
| if $key(f) \subseteq key(d) \subseteq key(e)$ | then | $\bar{g}(e) = \overline{key}(f)$ |
| in all remaining cases | | $\bar{g}(e) = \overline{key}(e)$ |

Note that $\bar{g}(e)$ is well-defined because from Lemma 7 it follows that any other triple of the form $d'ef'$ in $D$ such that $e$ is branching with $d', f'$ is such that $d' \sim d$ and $f' \sim f$. If $e$ is not branching then we define $\bar{g}(e) = \overline{key}(e)$. We also denote by $g(e)$ the set of elements occurring in the tuple $\bar{g}(e)$. From the definition we always have $g(e) \subseteq key(e)$.

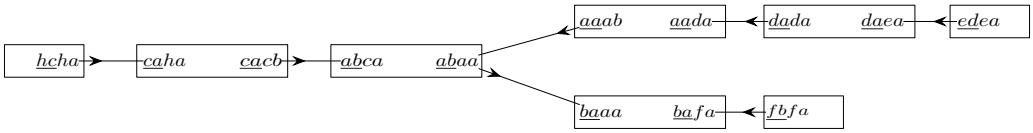
A **tripath** of $q$ is a database $\Theta$ such that each block $B$ of $\Theta$ contains at most two facts, and all the blocks of $\Theta$ can be arranged as a rooted tree with exactly two leaf blocks and satisfy the following properties (see Figure 1a). Let $s$ be the parent function between the blocks of $\Theta$ giving its tree structure: if $B$ is a block of $\Theta$ then $s(B)$ denotes the parent block of $B$. Then:




(a) Generic structure of a TRIPATH. The rectangles denote blocks and in every block $B$, $a(B)$ is denoted by a red dot and $b(B)$ is denoted by a blue dot. The root block has only $a(B)$ and leaf blocks have only $b(B)$. An undirected edge between two facts $s, t$ denotes that they form a solution $q\{st\}$. A directed edge from $s$ to $t$ denotes the solution $q(st)$. The unique branching fact of the TRIPATH is denoted by $e$ which forms a solution with the facts $d$ and $e$ with $q(de) \wedge q(ef)$. $def$ is the center of the TRIPATH. If the green solution $q(fd)$ is present then we call it a triangle-TRIPATH, otherwise it is a fork-TRIPATH. If the TRIPATH is not nice then there could be extra solutions to the query not depicted in the figure. The variable inclusion conditions are not depicted in the figure.




(b) An instance of a TRIPATH for the query $q = R(\underline{xu}\ xy) \wedge R(\underline{uy}\ xz)$. For the center of the given TRIPATH, $g(R(\underline{abaa})) = \{a\}$. The facts in the root and leaf blocks do not contain $a$ as a part of key. Note that there are extra solutions (in red) that are not enforced by the tripath. Hence this is not a nice-TRIPATH.




(c) An instance of a nice-TRIPATH for the query $q = R(\underline{xu}\ xy) \wedge R(\underline{uy}\ xz)$. We have the same center as in the previous case and $a$ does not belong to the key of the facts in root and leaf blocks. Note that there are no extra solutions other than those enforced by the TRIPATH.

Figure 1: Tripath illustrations

- There is exactly one block called the root block where the parent function $s$ is not defined and exactly two blocks, the leaf blocks, that have no children. Hence there is a unique block in $\Theta$, called the branching block, having two children.

- Let $B$ be a block of $\Theta$. If $B$ is the root block, it contains exactly one fact denoted by $a(B)$. If $B$ is one of the leaf blocks then $B$ contains exactly one fact denoted by $b(B)$. In all other cases, $B$ contains exactly two facts denoted by $a(B)$ and $b(B)$.

- Assume $B = s(B')$. Then $\Theta \models q\{a(B)\ b(B')\}$. In particular, for the branching block $B$ we have $e = a(B)$ which is a branching fact with $d = b(B')$ and $f = b(B'')$, where $B'$ and $B''$ are the two blocks whose parent is the branching block $B$. We call the triple $def$ as the **center** of the TRIPATH $\Theta$.

- Let $B_0$, $B_1$, $B_2$ be respectively the root and leaves of $\Theta$ and let $u_0 = a(B_0)$, $u_1 = b(B_1)$ and $u_2 = b(B_2)$. Let $B$ be the center block of $\Theta$ and $e = a(B)$. Then $g(e) \not\subseteq key(u_0)$ and $g(e) \not\subseteq key(u_1)$ and $g(e) \not\subseteq key(u_2)$.

We say that a database $D$ contains a TRIPATH of $q$ if there exists $\Theta \subseteq D$ such that $\Theta$ is a TRIPATH. A query $q$ admits a TRIPATH if there is a database instance $D$ of $q$ that contains a TRIPATH.

A TRIPATH $\Theta$ is called a fork-TRIPATH if the center facts $def$ of $\Theta$ forms a fork. If $def$ forms a triangle then $\Theta$ is called a triangle-TRIPATH.

The existence (or absence) of TRIPATH turns out to be the key in determining the complexity of the consistent query answering problem of the 2way-determined queries.

Notice that in the definition of TRIPATH we require the existence of some solutions to $q$ (namely $q\{a(B)b(B')\}$ where $B$ is the parent block of $B'$) but we do not forbid the presence of other extra solutions. In order to use the TRIPATH as a gadget for our lower bounds we need to remove those extra solutions. To this end we introduce a normal form for TRIPATH that in particular requires no extra solutions and show that if a TRIPATH exists then it exists in normal form.

For a TRIPATH $\Theta$, let $B_0$, $B_1$, $B_2$ be respectively the root and leaves of $\Theta$ and let $u_0 = a(B_0)$, $u_1 = b(B_1)$ and $u_2 = b(B_2)$. We say that $\Theta$ is **variable-nice** if there exists $x \in key(d), y \in key(e)$ and $z \in key(f)$ such that $\{x, y, z\} \cap \big(key(u_0) \cup key(u_1) \cup key(u_2)\big) = \emptyset$. We say that a TRIPATH is **solution-nice** if $q(\Theta) \subseteq \{\{ab\} \mid a = a(B_i), b = b(B_j), s(B_i) = B_j\} \cup \{\{fd\}\}$.

The variable-nice property identifies three elements one each from the key of the center facts $def$ such that the facts in the root and the leaf blocks do not contain these variables. To prove coNP-hardness, these variables will be used to the encoding. The solution-nice property ensures that $q$ holds in $\Theta$ only where it must hold by definition of being a TRIPATH, but nowhere else with the only exception of possibly $(fd)$, in which case $\Theta$ is a triangle-TRIPATH.

We say that a TRIPATH $\Theta$ is **nice** if the following holds:

- $\Theta$ is variable-nice

- $\Theta$ is solution-nice

- At least one of the elements of $x, y, z$ (from being variable-nice), appears in the key of all facts except $u_0$, $u_1$ and $u_2$.

- Each of the keys of $u_0$, $u_1$ and $u_2$ contains an element that does not occur in the key for any other facts in $\Theta$.

For instance the TRIPATH for $q_2$ depicted in Figure 1b is not nice since it contains some extra solutions. However Figure 1c depicts a nice TRIPATH for the same query $q_2$. It turns out that niceness can be assumed without loss of generality:

**Proposition 8.** *Let $q$ be a 2way-determined query. If $q$ admits a fork-*TRIPATH *(triangle-*TRIPATH*) then $q$ admits a nice fork-*TRIPATH *(triangle-*TRIPATH*).*

*Proof sketch.* Variable-niceness is achieved essentially by extending the branches of the TRIPATH depending on how $g(e)$ is defined. The construction of a solution-nice TRIPATH is more involved and is done by induction on the number of extra solutions. Typically, if $q(\alpha\beta)$ is an extra solution we will replace the fact $\alpha$ so that this extra solution is removed and add new blocks to the TRIPATH so that all other properties are satisfied. This can only work if $\alpha$ is not part of the center of the TRIPATH. When $\alpha$ is part of the center, it turns out that $\beta$ can not be part of the center. We then argue by symmetry using the block of $\beta$. The last two conditions are again simple to achieve. □

We will show in Section 8 that if a query $q$ does not admit a TRIPATH then CERTAIN($q$) can be solved in polynomial time using the greedy fixpoint algorithm of Section 5. If a query $q$ admits a fork-TRIPATH we will show in Section 9 that CERTAIN($q$) is coNP-complete. If a query $q$ does not admit a fork-TRIPATH but admits a triangle-TRIPATH we will show in Section 10 that CERTAIN($q$) can be solved in polynomial time, using a combination of the fixpoint algorithm of Section 5 and bipartite matching.

# 8 Queries with no tripath and PTIME

The main goal of this section is to prove that for every 2way-determined query $q$, if $q$ does not admit a TRIPATH then CERTAIN($q$) is in PTIME. There are many 2way-determined queries that have no TRIPATH. In Appendix E we give several sufficient syntactic conditions that imply this property. For instance the query $q_5 = R(\underline{x}\ yx)R(\underline{y}\ xu)$ does not admit a TRIPATH and hence CERTAIN($q_5$) is in PTIME which follows from the next theorem.

**Theorem 9.** *Let $q$ be a 2way-determined query. If $q$ does not admit a* TRIPATH *then* CERTAIN($q$) *is in* PTIME.

In fact we show that CERTAIN($q$) can be solved using the greedy fixpoint algorithm $\mathrm{Cert}_k(q)$ defined in Section 5 for $k = 2^{2\kappa+1} + \kappa - 1$ where $\kappa = l^l$ (recall that $l$ is the number of key positions in the relation $R$ under consideration)[4].

**Proposition 10.** *Let $q$ be a 2way-determined query and let $k = 2^{2\kappa+1} + \kappa - 1$ and let $D$ be a database. If $D$ does not admit a* TRIPATH *of $q$ then $D \in$ CERTAIN($q$) iff $D \in \mathrm{Cert}_k(q)$.*

For any database $D$ and repair $r$ of $D$ and $a \in r$ let r-key$(a, r) = \{c \mid c \in r \text{ and } key(c) \subseteq key(a)\}$. To prove Proposition 10 we build on the following lemma which resembles Lemma 6 but requires a more involved technical proof.

**Lemma 11.** *Let $k \geq \kappa$ and $q = AB$ be a query that is 2way-determined. Let $D$ be a database that does not admit a* TRIPATH. *Then for every repair $r$ of $D$ such that $r \models q\{ab\}$, and for all $K \subseteq r$ such that r-key$(a) \subseteq K$ and $|K| \leq k$, one of the following conditions hold:*

1. *$K \in \Delta_k(q, D)$*
2. *There exists a repair $r'$ such that $K \subseteq r'$ and $q(r') \subsetneq q(r)$.*

*Proof sketch.* Assume that $D$ does not admit a TRIPATH. Consider a repair $r$ of $D$, two facts $a, b$ of $r$ such that $r \models q\{ab\}$ and let $K$ be a set of facts containing r-key$(a, r)$. We need to show that if $K$ is not in $\Delta_k(q, D)$ then there is a new repair with strictly less solutions. We therefore need to remove at least one solution from $r$ and we have an obvious candidate as $r \models q\{ab\}$. We then use the definition of the algorithm $\mathrm{Cert}_k$ and from the fact that $K \notin \Delta_k(q, D)$ we know that in the block of $b$ there is a fact $b'$ that can not form a $k$-set when combined with facts from $K$. In particular $b'$ and $a$ do not form a solution to $q$. Let $r' = r[b \to b']$.

Now if $b'$ is not a part of any solution in $r'$ then $r'$ is the desired repair. Otherwise we can repeat the above argument with the facts that are making $q$ true when combined with $b'$. There are at most two such facts and we need to consider them both, one after the other. The goal is to

[4]Note that the constant $k$ directly results from the proof technique and is not intended to be optimal.

repeat the process above until all newly created solutions are removed from the working repair. When doing so we visit blocks of $D$, selecting two facts in each such block, the one that makes the query true with a previously selected fact, and the one we obtain from the non-membership to $\Delta_k(q, D)$. If we can enforce that we never visit a block twice we are done because the database being finite, eventually all the selected facts will not participate in a solution to $q$ in the current repair. In order to do this we keep in memory (the set $K$ initially) all the facts of the current repair that can potentially form a new solution. This ensures that we will never make the query true with a fact in a previously visited block. The difficulty is to ensure that the size of the memory remains bounded by $k$. This is achieved by requiring key inclusion between two consecutively selected facts, otherwise we stop and put a flag. If we get two flags we argue that we can extract a TRIPATH which is a contradiction. If not we can show that the memory remains bounded. □

We conclude this section with a sketch of the of proof of Proposition 10. We assume $D \models \text{CERTAIN}(q)$ and show that $D \models \text{Cert}_k(q)$. Let $r$ be a repair of $D$ that contains a minimal number of solutions. For any set of facts $K \subseteq r$ and $K' \subseteq D$, denote $K' \sim K$ if there is a bijection $f : K \to K'$ where $f(a) \sim a$ and let $r[K \to K']$ be the new repair obtained by replacing the facts of $K$ in $r$ by the facts of $K'$.

Since $D \models \text{CERTAIN}(q)$ there exists $a, b \in r$ such that $r \models q\{ab\}$. Let $K = \text{r-key}(a, r)$, clearly $|K| \leq \kappa$. It suffices to show that for all $K' \sim K$, $K' \in \Delta_k(q, D)$. If this is not the case for some $K'$, let $r' = r[K \to K']$. As $r$ is minimal, we can assume there are facts $c \in K'$ and $d \in r'$ such that $q\{cd\}$.

Notice that $\text{r-key}(c, r') \subseteq K'$. As $K' \notin \Delta_k(q, D)$, by Lemma 11, there exists a repair $r''$ such that $K' \subseteq r''$ and $q(r'') \subsetneq q(r')$. Repeating this argument eventually yields a repair contradicting the minimality of $r$.

# 9 Fork-tripath and coNP-hardness

In this section we prove that if a query that is 2way-determined admits a fork-TRIPATH, then CERTAIN($q$) is coNP-hard. We have already seen that the query $q_2 = R(\underline{xu}\ xy) \wedge R(\underline{uy}\ xz)$ admits a fork-TRIPATH (associated fork-TRIPATH are depicted in Figure 1 part (b) and (c)). In Appendix G we give several sufficient conditions that imply that a query admits a fork-TRIPATH. The fact that CERTAIN($q_2$) is coNP-hard is a consequence of the following result.

**Theorem 12.** *Let $q$ be a query that is 2way-determined. If $q$ admits a fork-*TRIPATH*, then* CERTAIN*($q$) is* coNP*-complete.*

The remaining part of this section is a proof of Theorem 12. In view of Proposition 8 we can assume that $q$ has a nice fork-TRIPATH $\Theta$. Let $x, y, z$ be the elements of $\Theta$ witnessing the variable-niceness of $\Theta$ and let $u, v, w$ be the fresh new elements occurring only in the keys of the head and tails of $\Theta$. Note that $x, y, z$ need not be distinct. For any elements $\alpha_x, \alpha_y, \alpha_z, \alpha_u, \alpha_v \alpha_w$, we denote by $\Theta[\alpha_x, \alpha_y, \alpha_z, \alpha_u, \alpha_v, \alpha_w]$ the database constructed from $\Theta$ by replacing each of $x, y, z, u, v, w$ by $\alpha_x, \alpha_y, \alpha_z, \alpha_u, \alpha_v, \alpha_w$ respectively, where $\alpha_x = \alpha_y$ iff $x = y$; $\alpha_y = \alpha_z$ iff $y = z$ and so on.

We use a reduction from 3-SAT where every variable occurs at most 3 times. Let $\phi$ be such a formula. Let $V_2$ be the variables of $\phi$ that occur exactly two times and $V_3$ be the variables of $\phi$ that occur exactly three times. Without loss of generality we can assume that each variable $p$ of $\phi$ occur at least once positively and at least once negatively. The construction is illustrated in Figure 2.

**The database.** Let $l \in V_3$. By our assumption, $l$ (or $\neg l$) occurs once positively - let $C[l]$ be this clause - and twice negatively - let $C_1[l]$, $C_2[l]$ be the two corresponding clauses.

Let $D[l]$ be the database consisting of the union of:
$\Theta_{l,C} = \Theta[\langle C, l\rangle_x, \langle C, l\rangle_y, \langle C, l\rangle_z, C, \langle C, C_2, l\rangle, \langle C, C_1, l\rangle]$
$\Theta_{l,C_1} = \Theta[\langle C_1, l\rangle_x, \langle C_1, l\rangle_y, \langle C_1, l\rangle_z, C_1, \langle C_1, C_1, l\rangle, \langle C, C_1, l\rangle]$ and
$\Theta_{l,C_2} = \Theta[\langle C_2, l\rangle_x, \langle C_2, l\rangle_y, \langle C_2, l\rangle_z, C_2, \langle C, C_2, l\rangle, \langle C_2, C_2, l\rangle]$.

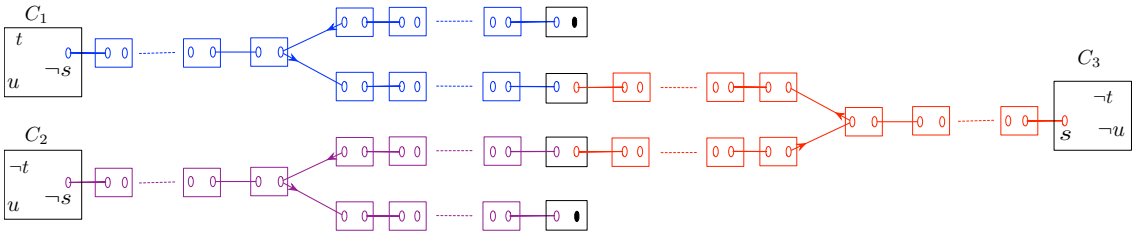



Figure 2: Consider the SAT formula $(\neg s \vee t \vee u) \wedge (\neg s \vee \neg t \vee u) \wedge (s \vee \neg t \vee \neg u)$. Each clause has a corresponding block denoted by $C_1, C_2$ and $C_3$ respectively. Each such block has three facts corresponding to the literals of the clause. The figure illustrates the gadget for the variable $s$. Similar construction is also done for the variables $t$ and $u$. Note that because the TRIPATH is solution-nice, if a repair $r$ makes the query false and picks $\neg s$ from $C_1$ and/or $C_2$ then it cannot have $s$ from $C_3$. Conversely if $r$ contains $s$ from $C_3$ then it cannot have $\neg s$ from both $C_1$ and $C_2$. The variable-niceness of the TRIPATH provides the necessary variables to encode the clause and the literals.

A few remarks about $D[l]$. The right leaf of $\Theta_{l,C}$ has the same key as the right leaf of $\Theta_{l,C_1}$ while the left leaf of $\Theta_{l,C}$ has the same key as the left leaf of $\Theta_{l,C_2}$. For the remaining blocks, the union is a disjoint union (because they contain $x, y$ or $z$ in $\Theta$, hence the element $\langle C, l\rangle_x, \langle C, l\rangle_y$ or $\langle C, l\rangle_z$ in $\Theta_{C,l}$ and so on). In particular all blocks have size two and each fact make the query true with a fact in a adjacent block.

Let now $l \in V_2$. By our assumption, $l$ (or $\neg l$) occur once positively - let $C[l]$ be this clause - and once negatively - let $C'[l]$ be the corresponding clause.

Let $D[l]$ be the database consisting of the union of
$\Theta_{l,C} = \Theta[\langle C, l\rangle_x, \langle C, l\rangle_y, \langle C, l\rangle_z, C, \langle C, C, l\rangle, \langle C, C', l\rangle]$ and
$\Theta_{l,C'} = \Theta[\langle C', l\rangle_x, \langle C', l\rangle_y, \langle C', l\rangle_z, C', \langle C', C', l\rangle, \langle C, C', l\rangle]$.

For the given 3-SAT formula $\phi$, define the corresponding database instance $D[\phi]$ as $\bigcup_{l\in\phi} D[l]$. Further, for every block $B$ in $D[\phi]$ if $B$ contains only one fact, then add a fresh fact in the block of $B$ that does not form a solution with any other facts of $D[\phi]$ (such a fact can always be defined for any block).

A few remarks about $D[\phi]$. Notice that by construction, every block of $D[\phi]$ has at least two facts. If $l$ and $l'$ occur in the same clause $C$ then the roots of $\Theta_{l,C}$ and $\Theta_{l',C}$ have the same keys. We call these heads the block of $C$ in the sequel. For all other blocks, the union of the $D[l]$ is a disjoint union of blocks as they all contain an element annotated with $l$ in their key. Consider now a pair of fact $a, b$ such that $D[\phi] \models q\{ab\}$. As the key of each element of $D[\phi]$ is annotated by either $C$ or $l$, $a$ and $b$ must belongs to the same $\Theta_{C,l}$ because $q$ is 2way-determined. Hence $a, b$ must be homomorphic copies of $a', b'$ in $\Theta$ such that $\Theta \models q(a'b')$. As $\Theta$ is solution-nice, $a', b'$ must be in consecutive blocks in $\Theta$.

The following lemma concludes the proof of Theorem 12.

**Lemma 13.** *Let $\phi$ be a 3-sat formula where every variable occurs at most three times. $\phi$ is satisfiable iff $D[\phi] \not\models$ CERTAIN$(q)$.*

# 10 Queries that admit only triangle-tripath

It remains to consider queries that admit at least one triangle-TRIPATH but no fork-TRIPATH. This is for instance the case for the query $q_6 = R(\underline{x}\ yz) \wedge R(\underline{z}\ xy)$ since $q_6$ does not have a fork-TRIPATH as all branching facts for $q_6$ form a triangle. However it is easy to construct a triangle-TRIPATH for $q_6$. A more challenging example is $q_7 = R(\underline{x_1x_2x_3}\ \underline{y_1y_1y_2y_3}\ \underline{z_1z_2z_3}\ z_4z_4z_4z_4) \wedge R(\underline{x_3x_1x_2}\ \underline{y_3y_1y_1y_2}\ \underline{z_2z_3z_4}\ z_1z_2z_3z_4)$. It is not immediate to construct a triangle-TRIPATH for $q_7$ and even less immediate to show that $q_7$ admits no fork-TRIPATH. This is left as a useful exercise to the reader.

We show in this section that for such queries, certain answers can be computed in polynomial

time. However, the following result shows that the greedy fixpoint algorithm of Section 5 does not work for such queries.

**Theorem 14.** *Let $q$ be a 2way-determined query admitting a triangle-*TRIPATH*. Then for all $k$,* CERTAIN$(q) \neq \text{Cert}_k(q)$.

The proof is essentially a reduction to the query $q_6$ for which it is shown in [FPSS23] that CERTAIN$(q_6)$ can not be solved using $\text{Cert}_k(q_6)$, for all $k$.

## 10.1 Bipartite matching algorithm

Since the algorithm $\text{Cert}_k(q)$ does not work, we need a different polynomial time algorithm to handle these queries. We use an algorithm based on bipartite matching, slightly extending the one introduced in [FPSS23] for self-join-free queries.

Note that since we only consider queries with two atoms, for every database $D$, it is convenient to describe the set of solutions to a query $q$ as a graph. We define the solution graph of $D$, denoted by $G(D, q)$ to be an undirected graph whose vertices are the facts of $D$ and there is an edge between two facts $a$ and $b$ in $G(D, q)$ iff $D \models q\{ab\}$. A connected component $C$ of $G(D, q)$ is called a **quasi-clique** if for all facts $a, b \in C$ such that $a \not\sim b$, $\{a, b\}$ is an edge in $G(D, q)$.

For an arbitrary database $D$, and each fact $a$ of $D$, *clique*$(a)$ is defined as follows : if $C$ is the connected component of $G(D, q)$ containing $a$ and $C$ is a quasi-clique *clique*$(a) = C$, otherwise *clique*$(a) = \{a\}$.

On input $D$, MATCHING(q) first computes $G(D, q)$ and its connected components, and then creates a bipartite graph $H(D, q) = (V_1 \cup V_2, E)$, where $V_1$ is the set of blocks of $D$ and $V_2 = \{\textit{clique}(a) \mid a \in D\}$. Further $(v_1, v_2) \in E$ iff the block $v_1$ contains a fact $a$ which is in $v_2$ and such that $D \not\models q(a, a)$. Note that constructing $G(D, q)$ and $H(D, q)$ can be achieved in polynomial time. Finally the algorithm outputs 'yes' iff there is a bipartite matching of $H(D, q)$ that saturates $V_1$. In this case we write $D \models$ MATCHING$(q)$. This can be checked in PTIME [HK73]. We now show that ¬MATCHING$(q)$ is always an under-approximation of CERTAIN$(q)$.

**Proposition 15.** *Let $q$ be a 2way-determined query and $D$ be a database. Then $D \models$* ¬MATCHING$(q)$ *implies $D \models$* CERTAIN$(q)$.

A database $D$ is called a **clique-database** for $q$ if every connected component $C$ of the graph $G(D, q)$ is a quasi-clique. As soon as the input database is a clique-database for $q$, ¬MATCHING$(q)$ correctly computes CERTAIN$(q)$ .

**Proposition 16.** *Let $q$ be a 2way-determined query and $D$ be a clique-database for $q$. Then $D \models$* ¬MATCHING$(q)$ *iff $D \models$* CERTAIN$(q)$. *Therefore checking whether $D \models$* CERTAIN$(q)$ *is in* PTIME.

This already gives the complexity of CERTAIN$(q)$ for some queries that do not admit fork-TRIPATH (but possibly admit triangle-TRIPATH). A query $q$ is said to be a **clique-query** if every database $D$ is a clique-database for $q$. For instance, the query $q_6$ is a clique-query as the solution graph of any database is a clique-database. From proposition 16 the following theorem follows.

**Theorem 17.** *Let $q$ be a 2way-determined query. If $q$ is a clique-query then* CERTAIN$(q)$ = ¬MATCHING$(q)$, *and thus* CERTAIN$(q)$ *is in* PTIME.

## 10.2 Combining matching-based and greedy fixpoint algorithms

In this section we prove that certain answers to a 2way-determined query $q$ which does not admit a fork-TRIPATH are computed by a combination of the polynomial time algorithms MATCHING$(q)$ and $\text{Cert}_k(q)$, thus completing the dichotomy classification (Recall that $\kappa = l^l$ where $l$ is the number of key positions).

**Theorem 18.** *Let $q$ be a query that is 2way-determined. If $q$ does not admit a fork-*TRIPATH *then* $\text{CERTAIN}(q) = \text{Cert}_k(q) \vee \neg\text{MATCHING}(q)$*, for* $k = 2^{2\kappa+1} + \kappa - 1$*. Thus* CERTAIN$(q)$ *is in polynomial time.*

The key to prove this theorem is the following proposition which proves that the database can be partitioned into components, such that on each component at least one of the two polynomial time algorithms is correct.

**Proposition 19.** *Let $q$ be a 2way-determined query that does not admit a fork-*TRIPATH *and let $D$ be a database. There exists a partition $C_1, C_2, \dots C_n$ of $D$ having all of the following properties :*

1. *for all $i$, $C_i$ does not contain a* TRIPATH *or $C_i$ is a clique-database for $q$.*
2. $D \models \text{CERTAIN}(q)$ *iff there exists some $i$ such that* $C_i \models \text{CERTAIN}(q)$.
3. *For all $k$, if* $C_i \models \text{Cert}_k(q)$ *for some $i$, then* $D \models \text{Cert}_k(q)$.
4. *If* $D \models \text{MATCHING}(q)$ *then for all $i$* $C_i \models \text{MATCHING}(q)$.

*Proof sketch.* The partition of the database is obtained using the following equivalence relation. Two blocks $B, B'$ of a database $D$ are said to be **$q$-connected** if $(B, B')$ belongs to the reflexive symmetric transitive closure of $\{(B_1, B_2) \mid \exists a \in B_1, b \in B_2 \text{ such that } D \models q\{ab\}\}$. The main difficulty is to show that each $q$-connected component satisfies (1) (the remaining properties are easy to show). If a $q$-connected component contains both a triangle-TRIPATH and a fork, then from Theorem 33 in Appendix G.2 it follows that $q$ admits a fork-TRIPATH. Hence each $q$-connected component is either a clique-database or contains no TRIPATH. □

*Proof of Theorem 18.* We assume Proposition 19 and prove the theorem. For an input database $D$, if $D \not\models \text{CERTAIN}(q)$ then, by Proposition 15, $D \models \text{MATCHING}(q)$; moreover $D \not\models \text{Cert}_k(q)$, as $\text{Cert}_k(q)$ too is always an under-approximation of CERTAIN$(q)$.

Assume now $D \models \text{CERTAIN}(q)$ and $D \models \text{MATCHING}(q)$, we show that $D \models \text{Cert}_k(q)$. Consider the partition $C_1, \dots C_n$ of $D$ given by Proposition 19. Since $D \models \text{CERTAIN}(q)$ there exists a $C_j$ such that $C_j \models \text{CERTAIN}(q)$; moreover $C_i \models \text{MATCHING}(q)$ for all $i$. In particular $C_j \models \text{MATCHING}(q)$ and therefore on $C_j$ the $\neg\text{MATCHING}(q)$ algorithm does not compute certain answers. Then by Proposition 16, $C_j$ is not a clique-database for $q$. Now by Proposition 19, $C_j$ admits no TRIPATH, and therefore by Proposition 10, $C_j \models \text{Cert}_k(q)$. Then, again by Proposition 19, $D \models \text{Cert}_k(q)$. □

# 11 Conclusion

We have proved the dichotomy conjecture on consistent query answering for queries with two atoms. The conditions we provided for separating the polynomial time case from the coNP-hard case can be shown to be decidable. Indeed one can show that if a fork-TRIPATH exists then there exists one of exponential size. However, it is likely that there are more efficient decision procedures than testing the existence of a TRIPATH.

We also obtained a (decidable) characterization of the two-atom queries whose certain answers are computable using the greedy fixpoint algorithm of Section 5 (under the standard complexity theoretic assumption that PTIME $\neq$ NP).

The dichotomy conjecture for all conjunctive queries remains a challenging problem. A new challenge that this paper poses is that of characterizing all conjunctive queries whose certain answers are computable by the greedy fixpoint algorithm of Section 5. We believe this is a worthwhile question, given the simplicity of this algorithm.

Work supported by ANR QUID, grant ANR-18-CE40-0031 and Padmanabha is funded by ANR-19-P3IA-0001 (PRAIRIE 3IA Institute)

# References


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

[AHV95] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[Ber19] Leopoldo E. Bertossi. Database repairs and consistent query answering: Origins and further developments. In Dan Suciu, Sebastian Skritek, and Christoph Koch, editors, *Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019*, pages 48–58. ACM, 2019.

[FPSS23] Diego Figueira, Anantha Padmanabha, Luc Segoufin, and Cristina Sirangelo. A simple algorithm for consistent query answering under primary keys. In *International Conference on Database Theory, ICDT*, 2023.

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

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

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

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

# A Proofs for Section 4 (First coNP-hard case)

**Proposition 2.** *Let $q$ be a query. There is a polynomial time reduction from* CERTAIN(*sjf*($q$)) *to* CERTAIN($q$).

*Proof.* Note that since $q = AB$ is not equivalent to a one-atom query we have $\overline{key}(A) \neq \overline{key}(B)$. Let $D$ be a database containing $R_1$-facts and $R_2$-facts. We construct in polynomial time a database $D'$ containing $R$-facts such that $D \models$ CERTAIN(*sjf*($q$)) iff $D' \models$ CERTAIN($q$).

For every $R_1$-fact $a$ of $D$, let $\mu(a)$ be a $R$-fact where for every position $i$, $\mu(a)[i]$ is the pair $\langle A[i], a[i]\rangle$. Similarly if $a$ is a $R_2$-fact then $\mu(a)$ is the $R$-fact such that $\mu(a)[i] = \langle B[i], a[i]\rangle$. Let $D' = \mu(D)$. Clearly $D'$ can be computed in polynomial time. Now we show that $D'$ has the desired properties.

Suppose $D' \models$ CERTAIN($q$). We show that all repairs of $D$ satisfy *sjf*($q$). Let $r$ be any arbitrary repair of $D$. Consider $r' = \mu(r)$. We claim that $r'$ is a repair of $D'$. To see this consider two facts $\mu(a), \mu(b) \in r'$. Suppose $\mu(a) \sim \mu(b)$, since $\overline{key}(A) \neq \overline{key}(B)$ this can only happen if $a \sim b$. But since $r$ is a repair, it follows that $a = b$ and therefore $\mu(a) = \mu(b)$. In order to conclude that $r'$ is a repair, it remains to show that any block of $D'$ has a representative in $r'$. Pick some block $B'$ in $D'$. Let $b' \in D$ such that $\mu(b') \in B'$ and let $b \in D$ such that $b \sim b'$ and $b \in r$. Then by construction $\mu(b) \in B'$ and we have $\mu(b) \in r'$. Now since $r'$ is a repair of $D'$, our hypothesis implies that $r' \models q(\mu(a)\mu(b))$, for some facts $a, b$ of $r$. If $a$ and $b$ are $R_1$ and $R_2$ facts then it immediately follows that $r \models$ *sjf*($q$)($ab$). If $a$ and $b$ are $R_2$ and $R_1$ facts respectively then it follows that $r \models$ *sjf*($q$)($ba$). Now suppose both $a$ and $b$ are $R_1$ facts then it follows that there is a homomorphism from $AB$ to $AA$. This implies that $q$ is equivalent to the one-atom query $R(A)$, a contradiction. Analogous contradiction is obtained if both $a, b$ are $R_2$ facts.

Suppose $D \models$ CERTAIN(*sjf*($q$)). We show that all repairs of $D'$ satisfy $q$. Let $r'$ be a repair of $D'$. Define $r = \{a \mid \mu(a) \in r'\}$. We first prove that $r$ is a repair of $D$. Towards this, consider two facts $a, b \in r$ such that $a \sim b$. Then by construction this implies $\mu(a) \sim \mu(b)$. As $r'$ is a repair this implies $a = b$. Now consider a block $B$ of $D$. Then $\mu(B)$ is a block of $D'$ and therefore has a representative $\mu(a)$ in $r'$. Hence $a$ is a representative of $B$ in $r$. As $r$ is a repair of $D$ there exists $a, b \in r$ such that $r \models$ *sjf*($q$)($ab$). It is immediate to check that $r' \models q(\mu(a)\mu(b))$. □

# B Proofs for Section 6 (First polynomial time case)

**Lemma 5.** *Let $q = AB$ be such that vars($A$) ∩ vars($B$) ⊆ key($B$) or key($A$) ⊆ key($B$). Then $q$ satisfies the zig-zag property.*

*Proof.* Let $D$ be a database, $a, b, b', c \in D$ such that $a \not\sim c$, $a \neq b$ and $b \sim b'$. It suffices to prove that there is a homomorphism from $AB$ to $ab'$. Clearly there is a homomorphism from $A$ to $a$ and from $B$ to $b'$. So, to prove that $D \models q(ab')$ it suffices to show that for any two positions $i, j$ if $A(i) = B(j)$ then $a(i) = b'(j)$. Pick any positions $i, j$ such that $A(i) = B(j)$. Then we have $a(i) = b(j)$ and $c(i) = b'(j)$.

If *vars*($A$) ∩ *vars*($B$) ⊆ *key*($B$) then there is a key position $j'$ such that $B(j) = B(j')$. This implies that $b'(j) = b'(j') = b(j') = b(j) = a(i)$.

Otherwise assume that *key*($A$) ⊆ *key*($B$). Since $a \not\sim c$, there is some key position $k$ such that $a(k) \neq c(k)$. By assumption there is some key position $k'$ such that $A(k) = B(k')$. This implies $b(k') \neq b'(k')$ which is a contradiction to $b \sim b'$. □

**Lemma 6.** *Let $q = AB$ be a query satisfying the zig-zag property. For all databases $D$ and for all repair $r$ of $D$ if $r \models q(ab)$ then $\{a\} \in \Delta_2(q, D)$ or there exists a repair $s$ of $D$ such that $q(s) \subsetneq q(r)$.*

To prove the lemma, we set up some definitions. For a database $D$, repair $r$ of $D$ and a fact $a \in r$ let $\mathrm{Sol}(a, r) = \{\ \{ab\} \mid r \models q\{ab\}\ \}$ and let $Sol(\overline{a}, r) = \{\ \{bc\} \mid r \models q\{bc\}$ and $b, c \neq a\ \}$. Note that $\mathrm{Sol}(a, r)$ and $Sol(\overline{a}, r)$ forms a disjoint partition of $q(r)$. Also we write $a \in \Delta_2(q, D)$ to mean if the set $\{a\}$ is in $\Delta_2(q, D)$.

*Proof.* Let $D$ be a database and $r$ be a repair of $D$. Let $a, b \in r$ such that $r \models q(ab)$. Now, assume that $a \notin \Delta_2(q, D)$. Then we build the repair $s$ that satisfies the required properties. To build $s$, we construct two sequences of facts $a_0, a_1, \ldots$ and $b_1, b_2, \ldots$ and a sequence of repairs $r_0, r_1, \ldots$ (by induction). Further, with $S_i = \{a_0, \ldots, a_i\}$ and $T_i = \{v \in r_i \mid \exists u \in S_i, r_i \models q(uv)\}$, we also maintain another sequence of facts $w_1, w_2, \ldots$ (possibly repeating) where each $w_i \in S_{i-1}$ such that the following invariants are satisfied for all $i$:

(a) $S_i \subseteq r_i$

(b) If $i > 0$ then $\forall v \in r_i$ we have $r_i \not\models q(va_i)$

(c) $\forall u, v \in S_i$ we have $r_i \not\models q(uv)$

(d) If $i > 0$ then the set $\{a, w_i, a_i\}$ does not contain any 2-set as a subset.

(e) If $i > 0$ then $b \notin r_i$

(f) if $i > 0$, then $b_i \in T_{i-1}$ and $a_i \sim b_i$ and $r_{i-1} \models q(w_i b_i)$

(g) $\{a, a_i\} \notin \Delta_2(q, D)$

(h) if $i > 0$, $\mathrm{Sol}(a, r_i) \subseteq \mathrm{Sol}(a, r) \setminus \{ab\}$

(i) if $i > 0$, $Sol(\overline{a}, r_i) \subseteq Sol(\overline{a}, r) \cup \{uv \mid u \in S_i, v \in T_i \text{ and } r_i \models q(uv)\}$

(j) $a_0, \ldots, a_i$ are pairwise distinct

In the base case define $a_0 = a$ and $r_0 = r$. Clearly (a), (g), (j) holds. Claim (c) holds because $a \notin \Delta_k(q, D)$. Claims (b), (d), (e), (f), (h), (i) are not applicable.

Inductively assume that the above properties are true for all $j \leq i$. Now, we either obtain the required $s$ or extend the sequences to $i + 1$ and show that the invariants are satisfied.

If $T_i = \emptyset$ then it implies that $i > 0$ (since $b \in T_0$). Hence, by (h), (i) it follows that $r_i$ is the desired repair .

If $T_i \neq \emptyset$, then if $i = 0$ then choose $b_1 = b$ and $w_1 = a$. Otherwise choose some $b_{i+1} \in T_i$ and let $w_{i+1} \in S_i$ be such that $r_i \models q(w_{i+1} b_{i+1})$. Note that $b_{i+1} \notin S_i$ (otherwise it contradicts (c)). Now, by (g) we have $\{a, w_{i+1}\} \notin \Delta_2(q, D)$. Hence there exists $a_{i+1} \sim b_{i+1}$ such that every subset of $\{a, w_{i+1}, a_{i+1}\}$ is not a 2-set. Set $r_{i+1} = r_i[b_{i+1} \to a_{i+1}]$.

We show that the inductive properties are satisfied for $i + 1$.

(a) This follows by construction since $b_{i+1} \notin S_i$ and $a_{i+1} \in r_{i+1}$.

(b) Assume, towards a contradiction, that there is some $v \in r_{i+1}$ such that $r_{i+1} \models q(va_{i+1})$. Since $D \models q(w_{i+1} b_{i+1})$, by zig-zag we have $D \models q(w_{i+1} a_{i+1})$. But this implies that $\{w_{i+1}, a_{i+1}\} \in \Delta_2(q, D)$ which is a contradiction to the construction.

(d) Immediate by construction.

(c) Observe that for all $u, v \in S_i \setminus S_{i+1}$, the claim follows by induction. So we only need to consider solutions that involve $a_{i+1}$. By (b) there is no $v \in S_{i+1}$ such that $r_{i+1} \models q(va_{i+1})$. We hence focus on the converse.

Towards a contradiction, assume that there is some $u \in S_{i+1}$ such that $r_{i+1} \models q(a_{i+1} u)$. Let $j$ be an index such that $u = a_j$. Since $\{a, a_{i+1}\} \notin \Delta_2(q, D)$ and $a = a_0 \in K$, we have $j > 0$. Hence by (f) there exists $b_j \in T_{j-1}$ such that $a_j \sim b_j$. Also there is some $w_j \in S_{j-1}$ such that $r_{j-1} \models q(w_j b_j)$. By zig-zag property we have $r_{j-1}[b_j \to a_j] \models q(w_j a_j)$ but this implies $\{w_j, a_j\}$ is a 2-set which contradicts (d).

(e) If $i = 1$ then we have $w_1 = a$ and $b_1 = b$ and construction $b \notin r_1$. Otherwise inductively $b \notin r_i$ so if $b \in r_{i+1}$ then the only possibility is $a_{i+1} = b$ but this implies $\{a, a_{i+1}\} \in \Delta_2(q, D)$ which is a contradiction.

(f) Immediate by construction.

(g) Immediate by construction.

(h) Pick some arbitrary $\{a, c\} \in \text{Sol}(a, r_{i+1})$. By (e) it follows that $c \neq b$.

Also if $\{a, c\} \in \text{Sol}(a, r_i)$ then inductively we have $\{a, c\} \in \text{Sol}(a, r)$. So the only possibility is $c = a_{i+1}$ but this implies $\{a, a_{i+1}\} \in \Delta_2(q, D)$ which is a contradiction.

(i) Pick some arbitrary $c'd' \in Sol(\overline{a}, r_{i+1})$ such that $r_{i+1} \models q(c'd')$. Suppose $c'd' \in Sol(\overline{a}, r)$, then by construction it has to be the case that either $c' \in S_{i+1}$ or $d' \in S_{i+1}$ (because all other facts are common to $r$ and $r_{i+1}$).

Now, if $c' \in S_{i+1}$ then by definition $d' \in T_{i+1}$ and hence $c'd' \in \{uv \mid u \in S_{i+1}, v \in T_{i+1}$ and $r_{i+1} \models q(uv)\}$. Otherwise if $d' \in S_{i+1}$ then it is a contradiction to (b) (since $d' \neq a$).

(j) Suppose not, then let $a_{i+1} = a_j$ for some $j \leq i$. By induction $a_j \in r_i$ and $b_{i+1} \sim a_{i+1}$ and also we have $r_i \models q(w_{i+1}b_{i+1})$. This implies $a_j = b_{i+1}$ and hence $r_i \models q(w_{i+1}a_j)$ which implies $w_{i+1} \in S_i$. This is in contradiction with (c).

Thus, all the items hold true for $i + 1$. Since $S_{i+1}$ strictly extends $S_i$ and the database is finite, at some point we must arrive at an index $j$ such that $T_j = \emptyset$, implying that $r_j$ is the required repair. □

# C Proofs for Section 7 (2way-determined queries)

**Proposition 8.** *Let $q$ be a 2way-determined query. If $q$ admits a fork-*TRIPATH *(triangle-*TRIPATH*) then $q$ admits a nice fork-*TRIPATH *(triangle-*TRIPATH*).*

*Proof.* Since $q = AB$ is 2way-determined, there exist variables $x_A \in key(A)$ and $x_B \in key(B)$ such that $x_A \in vars(B) \setminus key(B)$ and $x_B \in vars(A) \setminus key(A)$.

Let $\Theta$ be the TRIPATH admitted by $q$ centered at $def$ and let $u_e$ be the unique fact in the root block $U_e$ and $u_d$ and $u_f$ be the two facts in the two leaf blocks $U_d$ and $U_f$ respectively. We will obtain a new TRIPATH $\Theta'$ such that if $\Theta$ is a fork-TRIPATH (triangle-TRIPATH) then $\Theta'$ will be a nice fork-TRIPATH (nice triangle-TRIPATH).

Note that we have $g(e) \not\subseteq key(u_d)$, $g(e) \not\subseteq key(u_e)$ and $g(e) \not\subseteq (u_f)$. Assume that these non-key inclusions hold only at these three facts. So for all facts $u \in \Theta$ if $u \notin \{u_d, u_e, u_f\}$ we have $g(e) \subseteq key(u)$. Otherwise we can prune the branches of $\Theta$ making sure that this always holds.

We also assume that each of the key of $U_d$, $U_e$ and $U_f$ contains a unique element that does not occur as a key for any other facts in the TRIPATH. This can be achieved for $U_d$ (the other cases are treated similarly) as follows: Consider the block $B$ predecessor of $U_d$ in $\Theta$. Assume that $q(a(B)\ b(U_d))$ holds (the case where we have $q(b(U_d)a(B))$ is treated symmetrically). Let $\alpha$ be a fresh new element. Let $h$ be the morphism that maps $AB$ to $a(B)b(U_d)$. Let $h'$ be the morphism constructed from $h$ by mapping $x_B$ to $\alpha$. Let $a'b' = h'(AB)$ and notice that by our choice of $x_B$, $a' \sim a$ and $\alpha \in key(b')$ does not occur as key in any other facts. Let $U'_d$ be the new block that contains $b'$. It can also be verified that $key(b')$ does not contain all of $g(e)$ (otherwise this would also be true for $key(u_d)$). Construct the TRIPATH $\Theta'$ where $U_d$ is replaced by $U'_d$, $a(B)$ is replaced $a'$ and $s(B) = U'_d$.

Let $a_d \in key(u_d)$, $a_e \in key(u_e)$ and $a_f \in key(u_f)$ such that $a_d, a_e, a_f$ do not occur in any other key of any other facts of the TRIPATH $\Theta$. We start by making the TRIPATH $\Theta$ variable-nice. We do a case analysis depending on how $g(e)$ is defined.

1. if $key(d) \not\subseteq key(e)$ and $key(f) \not\subseteq key(e)$ then $g(e) = key(e)$. Let $B_e$ be the block of $e$.

   Hence we have $y_0, y_1, y_2 \in key(e)$ such that $y_0 \notin key(u_d)$ and $y_1 \notin key(u_e)$ and $y_2 \notin key(u_f)$. Our pruning on $\Theta$ implies that the variables $y_0, y_1$ and $y_2$ appear in the key of all other facts of $\Theta$ except $u_d, u_e, u_f$. Let $x \in key(d) \setminus key(e)$ and $z \in key(f) \setminus key(e)$.

In this case, to obtain variable-nice TRIPATH, the three elements that we pick are $x \in key(d)$ and $y = y_1 \in key(e)$ and $z \in key(f)$.

Note that $y \notin key(u_e)$. We can also assume that $x, z \notin key(u_e)$. This is because since $x \notin key(e)$ we can replace $x$ with a fresh new element $x'$ in all its occurrences in the path of $\Theta$ from $B_e$ to $U_e$, including $a(B_e)$ (but excluding $e$). This does not affect the *tripathness* of $\Theta$. We do similarly for $z$.

Consider now $u_d$. Let $\Theta_1$ be the path of $\Theta$ going from $u_d$ to $u_e$ (leaf to the root). Let $\Theta'_1$ be the copy of $\Theta_1$ after replacing $y_0$ by a fresh new element $y'_0$ and $a_e$ with a fresh element $a'_e$. Let $\hat{\Theta}_1$ be the database $\Theta_1 \cup \Theta'_1$ and replace $\Theta_1$ in $\Theta$ by $\hat{\Theta}_1$. This gives a new TRIPATH with the same $U_e, U_f$ but a new $\hat{U_d}$ instead of $U_d$. As $key(u_e) \cap \{x, y, z\} = \emptyset$ and $\hat{u}_d$ is a copy of $u_e$ then $key(\hat{u}_e) \cap \{x, y, z\} = \emptyset$. Also, we have $\hat{u_e} \not\sim \hat{u_d}$ since we have replaced $a_e$ with a fresh element $a'_e$.

By symmetry we do the same for $u_f$ (using $y_2$ instead of $y_0$ and $a''_e$ to substitute $a_e$ in the path from $u_f$ to $u_e$) and obtain a TRIPATH that is variable-nice.

2. if $key(d) \subseteq key(e)$ and $key(f) \not\subseteq key(e)$ then $g(e) = key(d)$.

   Then we have variables $x_0, x_1, x_2 \in key(d)$ such that $x_0 \notin key(u_d)$ and $x_1 \notin key(u_e)$ and $x_2 \notin key(u_f)$. Note that by assumption we have $x_0, x_1, x_2 \in key(e)$ and that they all appear in all blocks of $\Theta$ except for $u_d, u_e, u_f$. Also, let $z \in key(f) \setminus key(e)$. In this case the three elements that we pick are $x = x_0 = y$ and $z$.

   By construction $x \notin key(u_e)$ and as in the previous case we can make sure that $z \notin key(u_e)$.

   Consider now $u_d$. The same construction (using $x_1$ and $x_2$ instead of $y_1$ and $y_2$) as in the previous case shows that we can replace $u_d$ with $\hat{u}_d$ such that $\{x, z\} \cap key(\hat{u}_d) = \emptyset$.

   The case of $u_f$ is treated similarly.

3. if $key(d) \not\subseteq key(e)$ and $key(f) \subseteq key(e)$ then $g(e) = key(f)$.

   This case is symmetric to the previous case.

4. if $key(d) \subseteq key(f) \subseteq key(e)$ then $g(e) = key(d)$.

   Then we have variables $x_0, x_1, x_2 \in key(d)$ such that $x_0 \notin key(u_d)$ and $x_1 \notin key(u_e)$ and $x_3 \notin key(u_f)$. Note that by assumption we have $x_0, x_1, x_2 \in key(f) \subseteq key(e)$. In this case we use the element $x = y = z = x_0$. The rest of the construction is as above.

5. If $key(f) \subseteq key(d) \subseteq key(e)$ then in this case $g(e) = key(f)$.

   This case is symmetric to the previous case.

6. The only remaining case that is not covered before is when $key(d), key(f) \subseteq key(e)$ and $key(d) \not\subseteq key(f)$ and $key(f) \not\subseteq key(d)$. So, $g(e) = key(e)$.

   In this case we directly construct the variable-nice TRIPATH $\Theta'$ using the fork $def$ without using any other facts of $\Theta$. Let $x \in key(d) \setminus key(f)$ and $z \in key(f) \setminus key(d)$. By assumption we have $x, z \in key(e)$ and hence $z \in adom(d) \setminus key(d)$ and $x \in adom(f) \setminus key(f)$.

   We use the elements $x = y \in key(e) \cap key(d)$ and $z \in key(f)$ to construct $\Theta'$ and $def$ will be the center of $\Theta'$. The construction is depicted in Figure 3. We define the other facts of $\Theta'$ as follows:

   Let $h$ and $h'$ be the homomorphisms from $AB$ to $de$ and from $AB$ to $ef$ respectively.

   Let $Z$ be the set of variables $w$ of $AB$ such that $h(w) = z$ and let $\alpha$ be a fresh new element and let $h_1$ be the homomorphism such that $h_1(w) = \alpha$ if $w \in Z$ and $h_1(w) = h(w)$ otherwise. Let $d'b_1 = h_1(AB)$. From our choice of $z$ is follows that $d' \sim d$, $\alpha$ is part of the key of $b_1$ and $z$ does not appear in the key of $b_1$. Moreover for every position $i$, $b_1[i] \neq e[i]$ iff $e[i] = z$ and $b_1[i] = \alpha$. Hence there is a homomorphism from $e$ to $b_1$ (and hence from $A$ to $b_1$).

Now let $\beta$ be a fresh new element and let $h_2$ be the homomorphism such that for all $w \in vars(A) \cup vars(B)$ if $w = x_B$ then $h_2(w) = \beta$; if $w \in Z \setminus \{x_B\}$ then $h_2(w) = \alpha$ ; otherwise $h_2(w) = h'(w)$. Let $a_1 b_2 = h_2(AB)$. From our choice of $x_B$ it follows that $a_1 \sim b_1$ and $\beta$ is part of the key of $b_2$. From our choice of $x$ it follows that $x$ is not part of the key of $b_2$ (otherwise it can be argued that $x \in key(f)$ which is a contradiction). Altogether $d, d', a_1, b_1, b_2$ form a branch starting from the fact $d$ and ending to a fact $b_2$ that do not contain $x$ and $z$ in its key as desired.

Symmetrically we construct a branch starting from $f$ and ending to a fact that do not contain $x$ and $z$ in its key (using $X$ as the set of variables of $AB$ such that $h'(w) = x$ for every $w \in X$ for the first step and using $x_A$ instead of $x_B$ in the second step).

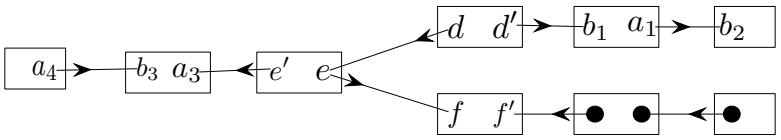



Figure 3: Illustration of the TRIPATH constructed in the special case when $key(d), key(f) \subseteq key(e)$ and $key(d) \not\subseteq key(f)$ and $key(f) \not\subseteq key(d)$.

It remains to construct the branch starting from $e$.

Let $\delta$ be a fresh new value and let $h_3$ be the homomorphism defined by $h_3(w) = \delta$ if $w = x_B$ and $h_3(w) = h'(w)$ otherwise. Let $e' a_3 = h_3(AB)$. From our choice of $x_B$ it follows that $e' \sim e$ and that $a_3$ contains $\delta$ in its key. It can also be verified that $x \notin key(a_3)$ (otherwise we can argue that $x \in key(f)$) but $x \in adom(a_3)$ (since $q$ is 2way-determined). Let $X'$ be the set of variables of $AB$ such that $h_3(w) = x$ for every $w \in X'$.

Let $\gamma$ be a fresh new value and let $h_4$ be the homomorphism defined by $h_4(w) = \gamma$ if $w \in X'$ and $h_4(w) = h_3(w)$ otherwise. Let $a_4 b_3 = h_4(AB)$. Because $x$ does not occur in the key of $a_3$ we have $b_3 \sim a_3$. Moreover $a_4$ contains $\gamma$ in its key.

Now we claim that there is a homomorphism from $\overline{key}(e)$ to $\overline{key}(a_4)$. To see this, pick any two key positions $i, j$ such that $e[i] = e[j]$. This implies that $e'[i] = e'[j]$ (since $e' \sim e$). Let $i', j'$ be positions such that $A(i) = B(i')$ and $A(j) = B(j')$ and hence $a_3[i'] = a_3[j']$. By construction this implies that $b_3[i'] = a_3[i'] = a_3[j'] = b_3[i']$. Hence we have $a_4[i] = b_3[i'] = b_3[j'] = a_4[j]$.

Thus there is a homomorphism from $\overline{key}(B)$ to $\overline{key}(a_4)$. Let $h_5$ be the homomorphism extending it to any variables of $AB$ by setting a new fresh values to all variables not in the key of $B$. Let $a_5 b_4 = h_5(AB)$. It can be verified that $z$ is no longer in the key of $a_5$ as desired (otherwise we can argue that $z \in key(d)$).

This concludes the proof that we can obtain a variable-nice TRIPATH $\Theta$. We now show that we can further enforce solution-niceness.

We say that a solution $q(\alpha\beta)$ is good in the TRIPATH $\Theta$ if both $\alpha, \beta \in \{d, e, f\}$ or $\{\alpha, \beta\} = \{a(B), b(B')\}$ for some blocks $B, B'$ where $B$ is the parent of $B'$ in $\Theta$. Otherwise, $q(\alpha\beta)$ is called an extra solution. If all solutions of $\Theta$ are good then $\Theta$ is already a solution-nice. Otherwise we strictly decrease the number of extra solutions within $\Theta$ by transforming it. As the transformation will preserve variable-niceness, this eventually yields a solution-nice and variable-nice TRIPATH.

Consider an extra solution $q(\alpha\beta) \in q(\Theta)$.

Notice that because $\alpha$ and $\beta$ are part of a TRIPATH, they must be part of some good solutions. Let $c_1$ and $c_2$ be the facts of $\Theta$ such that $q\{c_1\alpha\}$ and $q\{c_2\beta\}$ are good solutions. Notice that because $q$ is 2way-determined, if $q(\alpha c_1)$ then $c_1 \sim \beta$ and if $q(c_2\beta)$ then $\alpha \sim c_2$ by Lemma 7.

We consider several cases.

**Case 1**. Assume first that $q(\alpha c_1)$ holds (hence $c_1 \sim \beta$). Notice that $c_2 \not\sim \alpha$ otherwise this would break the tree structure $\Theta$. This implies that $q(\beta c_2)$. Notice that $\beta \notin \{ef\}$ as $q(\delta\beta)$ (for any arbitrary fact $\delta$) would imply $\delta \sim \alpha$ and there would be two good edges from the block of $\alpha$ to the block of $\beta$. If $\beta = d$ (hence $c_2 = e$), notice that $\alpha \notin \{def\}$ and $c_1 \notin \{def\}$. Similarly we can

argue that if $c_2 \in \{def\}$ then $c_2 = e$ and $\beta = d$. We therefore distinguish between two subcases depending on whether $\beta = d$ or not.

**Subcase 1.1**. If $\beta = d$ then $c_2 = e$. Hence $\alpha$ is not a part of the center of the TRIPATH $\Theta$. So we can safely replace $\alpha$ and $c_1$ as follows: Let $h$ be the homomorphism from $AB$ to $\alpha c_1$. Let $h_1$ be the one constructed from $h$ by setting $x_A$ to a fresh new value $\hat{x}_A$. Set $a_1 c_1' = h_1(AB)$. Our choice of $x_A$ implies that $c_1' \sim c_1$ and $a_1$ is in a fresh new block.

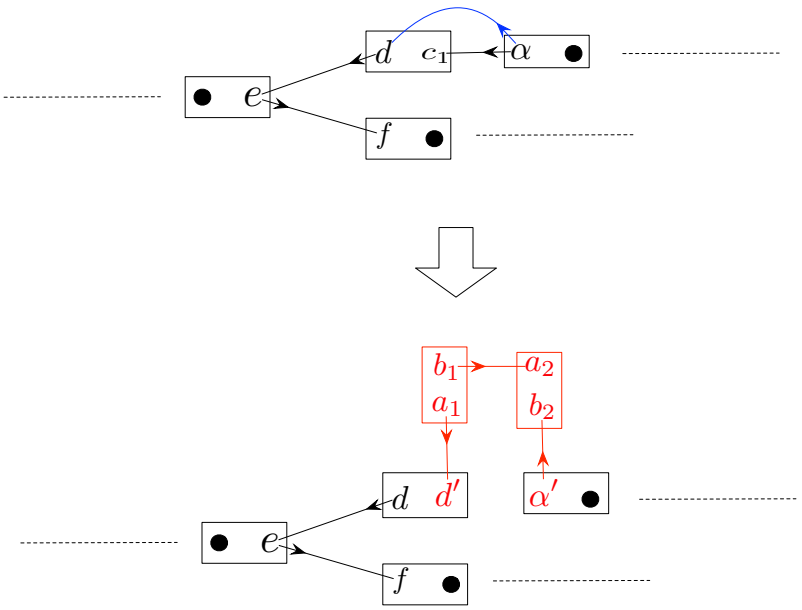



Figure 4: Illustration of the construction making the TRIPATH solution nice in Case 1.1. The red part are the new added facts

Let $h_2$ be the homomorphism constructed from $h$ by setting $x_A$ to $\hat{x}_A$ and $x_B$ to a fresh new value $\hat{x}_B$. Let $b_1 a_2 = h_2(AB)$. Our choice of $x_B$ implies that $b_1 \sim a_1$ and $a_2$ is in a fresh new block.

Finally, let $h_3$ be the homomorphism constructed from $h$ by setting $x_B$ to $\hat{x}_B$. Let $\alpha' b_2 = h_2(AB)$. Our choice of $x_B$ implies that $b_2 \sim a_2$ and $\alpha \sim \alpha'$.

We construct $\Theta'$ by replacing $\alpha$ by $\alpha'$, $c_1$ by $c_1'$ and adding the facts $a_1, b_1, a_2, b_2$ with appropriate blocks and their successors. By construction $def$ remains the center of $\Theta'$ and $\Theta$ remains variable-nice. We need to argue about the extra solutions. Notice that each of the $a_i$ and $b_i$ can not be part of an extra solution because they have a fresh new values in their key that do not occur in any other fact but the one that already form a good solution with them. Consider $\alpha'$. Assume $q(\alpha'\delta)$ then $\delta \sim c_1$ so $\delta = c_1'$. But then $\alpha' \sim a_1$ and this is not true by construction. Assume now $q(\delta\alpha')$. We can argue that this implies $q(\delta\alpha)$ (since for every position $i$, if $\alpha[i] \neq \alpha'[i]$ then $\alpha[i]$ is fresh). Hence such extra solutions are already present in $\Theta$ and we do not increase the number of extra solutions. Finally consider the fact $c_1'$. Assume $q(\delta c_1')$ then $\delta \sim a_1$ and so $\delta = b_1$. But then $c_1' \sim a_2$ and this is not true by construction. Assume now $q(c_1'\delta)$. We can argue that this implies $q(c_1\delta)$ and hence we do not increase the number of extra solutions..

The construction is described in Figure 4. Altogether the number of extra solutions has decreased by one.

**Subcase 1.2**. Assume now that $\beta \neq d$ and $c_2 \neq e$. As we have seen, this implies that $\beta, c_2 \notin \{def\}$. We can safely replace $\beta$ and $c_2$ as follows:

We construct a new TRIPATH $\Theta'$, with the same center $def$, replacing $\beta$ and $c_2$ by new facts $\beta'$ and $c_2'$ in their respective blocks and linking them by new solutions to $q$ in order to make sure that $q(\alpha\beta')$ does not hold. The construction is such that no new extra solution is created.

To achieve this, by Corollary 27 we can assume that the generalized 2-path of $q$ is such that

$key(P_1) \not\subseteq key(P_0)$ and $key(P_1) \not\subseteq key(P_2)$.

Let $i_1$ be a key position where $P_1[i_1] = y_1 \notin key(P_0)$. This implies that $B[i_1] \notin key(A)$ and since $q$ is 2way-determined, there is a non-key position $j_1$ such that $B[i_1] = A[j_1]$. Let $P_1'[j_1] = z$ and by construction, $z \notin vars(P_1)$ and $P_2[i_1] = z \in key(P_2)$.
Similarly let $i_2$ be the key position where $P_1[i_2] = y_2 \notin key(P_2)$. This implies that $A[i_2] \notin key(B)$ and by definition of the generalized 2 path, $P_0[i_2] = x \notin key(P_1)$.

Thus we have variables $x, y_1, y_2, z$ such that $x \in key(P_0) \setminus key(P_1)$; $y_1 \in key(P_1) \setminus key(P_0)$; $y_2 \in key(P_1) \setminus key(P_2)$ and $z \in key(P_2) \setminus key(P_1)$ where $z \notin vars(P_1)$ (Recall that $key(P_1) = key(P_1')$). We will use these variables to modify the facts of the TRIPATH to remove extra solutions.

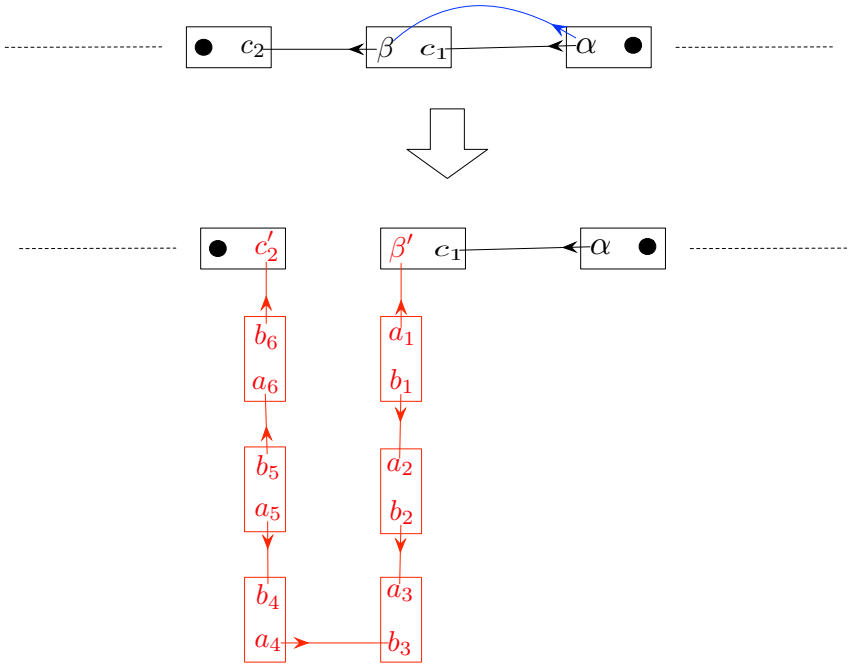



Figure 5: Illustration of the construction making the TRIPATH solution nice in Case 1.2. The red part are the new added facts

By construction there is a homomorphism $h$ from $(P_0P_1P_1'P_2)$ to $\alpha c_1 \beta c_2$.

Let $h_1$ be the morphism constructed from $h$ by setting $x$ to a fresh element $\hat{x}$ not occurring in $\Theta$. Let $a_1\beta' = h_1(P_0P_1)$. Notice that by our choice of $x$, $\beta' \sim \beta$ (since $x \notin key(P1)$), while $a_1$ is a fact of a new block $B_1$ (since $x \in key(P_0)$).

Let $h_2$ be the morphism constructed from $h$ by setting $h_2(x) = \hat{x}$, $h_2(y_1) = \hat{y}_1$ and $h_2(z) = \hat{z}$, where $\hat{y}_1$ and $\hat{z}$ are fresh new values. Let $b_1a_2b_2a_3 = h_2(P_0P_1P_1'P_2)$. By construction we have $a_i \sim b_i$ for $i = 1, 2$ and $a_2$ and $a_3$ contain a fresh new value in their key.

Let $h_3$ be the morphism constructed from $h$ by setting $h_3(z) = \hat{z}$, $h_3(y_1) = \hat{y}_1$, $h_3(y_2) = \hat{y}_2$ and $h_3(x) = \dot{x}$ where $\dot{x}$ and $\hat{y}_2$ are fresh new values. Let $a_5b_4a_4b_3 = h_3(P_0P_1P_1'P_2)$. By construction we have $a_i \sim b_i$ for $i = 1, 2, 3, 4$ and $a_4$ and $a_5$ contain a fresh element in their key.

Finally let $h_4$ be the morphism constructed from $h$ by setting $h_3(x) = \dot{x}$, $h_3(y_2) = \hat{y}_2$ is a fresh new value. Let $b_5a_6b_6c_2' = h_3(P_0P_1P_1'P_2)$. By construction we have $a_i \sim b_i$ for $i = 1, 2, 3, 4, 5, 6$ and $a_6$ contains a fresh new value in their key. Moreover we have $c_2' \sim c_2$.

Let $\Theta'$ be the database constructed from $\Theta$ by replacing $\beta$ with $\beta'$, $c_2$ with $c_2'$ and adding the facts $a_1b_1a_2b_2 \cdots a_6b_6$ in appropriate blocks between the block of $\beta$ and the block of $c_2$ and modifying the successor function for blocks appropriately.

By construction the center of $\Theta'$ is $def$ and hence $\Theta'$ is variable-nice. It is clear that all the new blocks have size 2, containing $a_i$ and $b_i$ respectively.

It remains to consider the extra solutions. As above we can argue than none of them can involve one of the $a_i$ or the one of the $b_i$. Consider $\beta'$. Assume $q(\delta\beta')$. Then $\delta \sim a_1$ and therefore

$\delta = b_1$ and we have seen that this is not possible. Assume $q(\beta'\delta)$. Then we can argue that $q(\beta\delta)$, so such solutions do not increase the number of extra solutions in the TRIPATH. Consider $c_2'$. Assume $q(\delta c_2')$. Then $\delta \sim b_6$, so $\delta = a_6$ and we have seen that this is can not be the case. Assume $q(c_2'\delta)$. Then we can argue that $q(c_2\delta)$ also hold.

The construction is described in Figure 5. Altogether the number of extra solution has decreased by one.

**Case 2**. The case where $q(c_2\beta)$ is handled as the previous case by symmetry.

**Case 3**. Assume now $q(c_1\alpha)$ and $c_1, \alpha \notin \{def\}$.

We argue as in Case 1.1 for replacing $\alpha$ and $c_1$ with new facts while inserting two new blocks.

**Case 4**. The case where $q(\beta c_2)$ and $c_1, \alpha \notin \{def\}$ is treated similarly as Case 3.

**Case 5**. It remains to consider the case where $q(c_1\alpha)$, $q(\beta c_2)$ and one of $c_1, \alpha$ is in $\{def\}$ and one of $c_2, \beta$ is in $\{def\}$. A simple case analysis shows that this can only happen when $\alpha = f$ and $\beta = d$, but then $q(\alpha\beta)$ is not an extra solution. This only means that $\Theta$ was a triangle-TRIPATH. □

# D Proofs for Section 8 (Queries with no tripath and PTIME)

**Proposition 10.** *Let $q$ be a 2way-determined query and let $k = 2^{2\kappa+1} + \kappa - 1$ and let $D$ be a database. If $D$ does not admit a* TRIPATH *of $q$ then $D \in$ CERTAIN$(q)$ iff $D \in \text{Cert}_k(q)$.*

*Proof.* Let us assume Lemma 11 and prove the proposition. If $D \in \text{Cert}_k(q)$ then clearly $D \in$ CERTAIN$(q)$ since $\text{Cert}_k(q)$ is always an under-approximation of CERTAIN$(q)$. So we only need to prove the other direction. Assume $D \models$ CERTAIN$(q)$. We need to prove that $D \models \text{Cert}_k(q)$. Let $r$ be a repair of $D$ that contains minimal number of solutions.

Since $D \models$ CERTAIN$(q)$ there exists $a, b \in r$ such that $r \models q\{ab\}$. Let $K = \text{r-key}(a, r)$, clearly $|K| \leq \kappa$. Now we claim that for all $K' \sim K$ we have $K' \in \Delta_k(q, D)$. Then by the update rule of the algorithm it follows that $\emptyset \in \Delta_k(q, D)$ and we are done.

Suppose the claim is false. Then let $K' \sim K$ such that $K' \notin \Delta_k(q, D)$. Let $r'$ be a repair such that $K' \in r'$. Let $a' \sim a$ such that $a' \in K'$. Define $r_0 = r[K \to K']$. Note that $\text{r-key}(a', r_0) = K'$ and $|K'| = |K| \leq \kappa$. Let $|q(r_0)| = l$. Then $l \geq |q(r)|$ because repair $r$ is minimal.

We construct a sequence of repairs $r_0, r_1, \ldots r_l$ such that for all $i \leq l$ we have $q(r_i) \subsetneq q(r_{i-1})$. This implies that $|q(r_l)| \leq 0$ which is a contradiction to $D \models$ CERTAIN$(q)$). The sequence of repairs is constructed by induction such that for all $i \leq l$ the following invariants are maintained:

(a) $K' \subseteq r_i$

(b) if $i > 0$ then $q(r_i) \subsetneq q(r_{i-1})$

We have already defined $r_0$ for which $(a)$ holds by construction.

For the induction step, assume that we have constructed $r_0, \ldots r_i$ for some $i < l$ satisfying the inductive invariants.

Assume first there are not fact $a \in K'$ such that there is a fact $b \in r_i$ such that $q\{ab\}$ holds. Then $q(r_i) \subseteq q(r) \setminus \{(ab)\}$, a contradiction with the minimality of $r$.

Therefore there are facts $a_i, b_i$ such that $a_i \in K'$ and $b_i \in r_i$ such that $q\{a_ib_i\}$ holds. Since $K' \subseteq r_i$ then we have $\text{r-key}(a_i, r_i) \subseteq K'$. We also have $|K'| \leq \kappa$ and by assumption $K' \notin \Delta_k(q, D)$. Hence, by lemma 11, there exists a repair $r_{i+1}$ such that $K' \subseteq r_{i+1}$ and $q(r_{i+1}) \subsetneq q(r_i)$ as desired. □

Given a set of facts $K'$ with $K' \subseteq D$, the *active* facts of $K'$ are all facts $c \in K'$ such that $D \models q\{cc'\}$ for some fact $c'$ of $D$ such that for all $f \in K'$, $c' \not\sim f$.

**Lemma 11.** *Let $k \geq \kappa$ and $q = AB$ be a query that is 2way-determined. Let $D$ be a database that does not admit a* TRIPATH. *Then for every repair $r$ of $D$ such that $r \models q\{ab\}$, and for all $K \subseteq r$ such that r-key$(a) \subseteq K$ and $|K| \leq k$, one of the following conditions hold:*

1. $K \in \Delta_k(q, D)$

*2. There exists a repair $r'$ such that $K \subseteq r'$ and $q(r') \subsetneq q(r)$.*

To prove the lemma we will use the following result.

**Lemma 20.** *Let $D, K, r, a, b$ be as described in Lemma 11. If there exists $K'$ with $K \subseteq K' \subseteq D$, such that the facts of $K'$ have pairwise distinct keys, $q(K')$ is empty and, all active facts of $K'$ are in $K \setminus \{a\}$, then there exists a repair $r'$ of $D$ such that $K \subseteq r'$ and $q(r') \subsetneq q(r)$.*

*Proof.* Since facts of $K'$ have pairwise distinct keys, let $K_1 \subseteq r$ such that $K_1 \sim K'$. Define $r' = r[K_1 \to K']$. Clearly $r'$ contains $K$ since $K \subseteq K'$. Now we verify that $q(r') \subsetneq q(r)$. Suppose $r' \models q(fg)$ for some facts $f$ and $g$.

If $f \in K'$, then $g \in r' \setminus K'$ since $q(K') = \emptyset$. Thus $f$ is an active fact of $K'$ and hence $f \in K \setminus \{a\}$. Moreover $g \in r' \setminus K'$ implies $g \in r$ and hence $r \models q(fg)$. By symmetry we get the same result if $g \in K'$. If neither $f$ nor $g$ are in $K'$, then they are both $f, g \in r$ and therefore $r \models q(fg)$. Altogether this shows that $q(r') \subseteq q(r)$. The inclusion is strict because $a$ is not an active fact and hence $r' \not\models q\{ab\}$. □

To prove Lemma 11, let $D$ be a database such that $D$ does not admit a TRIPATH. Pick a repair $r$ of $D$. Let $r \models q\{ab\}$ and let $K \subseteq r$ where r-key$(a, r) \subseteq K$ and $|K| \leq \kappa$. Assume that $K \notin \Delta_k(q, D)$; we construct the desired repair $r'$.

Towards this, from Lemma 20 it suffices to show that a set $K'$ as described in Lemma 20 exists. We construct $K'$ by induction, by increasingly adding one new element $a_i$ at a time. In order for the induction to work, the intermediate sets $K \cup \{a_0, ...a_n\}$ must satisfy some extra properties than the ones needed for $K'$. These properties are formalised by the following lemma.

**Lemma 21.** *Let $D$ and $K$ be as described in Lemma 11. Assume there exists a sequence of distinct blocks $B_0, \cdots, B_n$ of $D$ satisfying the following properties : $B_0$ is the block of $a$ and, for all $j > 0$, $B_j$ contains two distinct facts $a_j, b_j$ and no fact from $K$. Moreover let $a_0 = a$, let $K_n = \{a_0, \cdots, a_n\} \cup K$, let $A_n = \{a_j \mid j \leq n$ and $a_j$ is an active fact of $K_n\}$, and let $\mathcal{A}_n = \{B_j | a_j \in A_n\}$. Additionally assume the following set of properties, denoted by $C_n$:*

*(a) $B_0, \cdots, B_n$ form a binary tree $\mathcal{T}_n$, whose parent function is denoted by $s_n$, and whose root is $B_0$. Further, whenever $B_i = s_n(B_j)$ then $D \models q\{a_i b_j\}$.*

*(b) Let $\mathcal{F}_n$ be the set of all $B_i$ having two distinct children in $\mathcal{T}_n$, both having a descendant belonging to $\mathcal{A}_n$. Then for each $B_l \in \mathcal{A}_n \cup \mathcal{F}_n$, $l > 0$, and each $B_j$ which is a non-leaf descendant of $B_l$ in $\mathcal{T}_n$, one has $g(a_l) \subseteq key(a_j)$.*

*(c) For each $B_i \in \mathcal{A}_n \cup \mathcal{F}_n$ which is not a leaf of $\mathcal{T}_n$, there exists $B_j$ such that $\bar{g}(a_i) = \overline{key}(a_j)$. Moreover either $B_i = B_j$ or $B_i = s_n(B_j)$.*

*(d) $|A_n \cup K| \leq k$ and $A_n \cup K \notin \Delta_k(q, D)$.*

*(e) $q(K_n) = \emptyset$.*

*Then there exists a set $K'$ with $K_n \subseteq K' \subseteq D$, whose facts have pairwise distinct keys, containing no solution, and whose only active facts are in $K \setminus \{a\}$.*

Note that $q(K) = \emptyset$, otherwise $K \in \Delta_k(q, D)$. Hence for the singleton sequence $B_0$, which is the block of $a$, satisfies the hypotheses of Lemma 21, with $K_0 = K$, $A_0 = \{a\}$, $\mathcal{A}_0 = \{B_0\}$ and $\mathcal{F}_0 = \emptyset$. So Proposition 10 follows from Lemma 21.

*Proof of Lemma 21.* First notice that all active fact of $K_n$ are either in $A_n$ or in $K \setminus \{a\}$. Therefore, whenever $A_n = \emptyset$ one can take $K' = K_n$ to obtain the desired properties. Our aim is to extend $K_n$ until $A_n = \emptyset$.

We prove the lemma by induction on $n$. The limit case is when $n$ is maximal (given by the difference between the number of blocks in $D$ and $|K|$), we have $A_n = \emptyset$ since all the keys of $D$ occur in $K_n$. Then, as observed above, $K' = K_n$ is the desired set.

We now assume that the lemma holds for $n \geq 0$, we prove it for $n+1$. Note that if $n = 0$ then clearly $A_n \neq \emptyset$. If $n \geq 1$ and $A_n = \emptyset$ we take $K' = K_n$ and conclude. So assume $C_n$ holds for a sequence of blocks $B_0, \dots B_n$ and $A_n \neq \emptyset$. We show how to find a new block $B_{n+1}$ such that $B_0, \dots B_n, B_{n+1}$ satisfies $C_{n+1}$. Then, by applying the induction hypothesis, one can construct $K'$ with $K_n \subseteq K_{n+1} \subseteq K' \subseteq D$ whose facts have pairwise distinct keys, containing no solution, and whose only active facts are in $K \setminus \{a\}$; this will complete the proof.

We will be adding a new block $B_{n+1}$ to the sequence as a child of some node $B_{j^*}$ of $\mathcal{T}_n$. As $B_{j^*}$ then becomes non-leaf, we have to make sure that the inclusions of Item (b) are satisfied for $B_{j^*}$. We show that a node $B_{j^*}$ with this property must exist. More precisely we prove that :

**Claim 22.** *There exists a block $B_{j^*} \in \mathcal{A}_n$ such that for all block $B_l \in \mathcal{A}_n \cup \mathcal{F}_n$, with $l > 0$ and $B_l$ ancestor of $B_{j^*}$ in $\mathcal{T}_n$, one has $g(a_l) \subseteq key(a_{j^*})$.*

Towards proving the claim, we will use the following consequence of the fact that $D$ does not admit a TRIPATH:

**Fact 23.** For all blocks $B_l \in \mathcal{F}_n$ with $l > 0$, there exists at least one child of $B_l$ in $\mathcal{T}_n$ such that for all blocks $B_j$ in its subtree (including its leaves), $g(a_l) \subseteq key(a_j)$.

To see this, assume that this is not the case for some $B_l$. Since $B_l \in \mathcal{F}_n$, there are two children on $B_l$ in $\mathcal{T}_n$. So there exists a block $B_{l_1}$ in the right subtree of $B_l$ and a block $B_{l_2}$ in its left subtree with $g(a_l) \not\subseteq key(a_{l_1})$, and $g(a_l) \not\subseteq key(a_{l_2})$. By Item (c) $g(a_l) = key(a_j)$, for some $B_j$ descendant of $B_l$ and by hypothesis $key(a_j)$ can not be included into $key(f)$ for some fact $f \in K$, in particular $key(a_j) \not\subseteq key(a)$ (otherwise $a_j \in K$). Hence we also have $g(a_l) \not\subseteq key(a_0)$. Therefore the tree formed by $B_{l_1}$, $B_{l_2}$ and all their ancestors contains a TRIPATH (by keeping only the fact $a_i, b_i$ in all blocks $B_i$ of this tree), which is a contradiction.

Now the desired block $B_{j^*}$ for Claim 22 is given by the following procedure: Starting from the root of $\mathcal{T}_n$ we follow a path visiting only nodes having a descendant in $\mathcal{A}_n$, and we stop as soon as we find a node in $\mathcal{A}_n$. (notice that the root has a descendant in $\mathcal{A}_n$ since $\mathcal{A}_n \neq \emptyset$). While the current node is not in $\mathcal{A}_n$ it must have a child with a descendant in $\mathcal{A}_n$. If the current node is the root or is not in $\mathcal{F}_n$ we move to any such child. Otherwise (the current node is in $\mathcal{F}_n$ and is not the root) we move to its child satisfying Fact 23. When we stop, we are on a node $B_{j^*} \in \mathcal{A}_n$ having no proper ancestor in $\mathcal{A}_n$, and such that all its ancestors $B_l \in \mathcal{E}_n$ with $l > 0$ satisfy $g(a_l) \subseteq key(a_{j^*})$. Moreover by definition of the function $g$, we have $g(a_{j^*}) \subseteq key(a_{j^*})$, thus $B_{j^*}$ is the desired block.

Let $B_{j^*}$ be the block given by Claim 22. Since $a_{j^*} \in A_n$, there exists a fact $c$ such that $D \models q\{a_{j^*}c\}$ and $\overline{key}(c)$ does not occur in $K_n$. If there are two facts $c$ with different keys having this property, and if moreover $\bar{g}(a_{j^*}) \neq \overline{key}(a_{j^*})$, then we choose a $c$ so that $\overline{key}(c) = \bar{g}(a_{j^*})$ (which must exist by the definition of $\bar{g}$). By Item (d) we know that $A_n \cup K \notin \Delta_k(q, D)$ and therefore there exists a fact $c' \sim c$ such that $c' \cup A_n \cup K$ contains no $k$-sets.

Let $b_{n+1} = c$, $a_{n+1} = c'$ and $B_{n+1}$ be the block of $c$. Let $s_{n+1}(B_{n+1}) = B_{j^*}$ and let $s_{n+1}$ coincide with $s_n$ on $B_0, \dots B_n$ (*i.e.* $T_{n+1}$ is obtained by appending $B_{n+1}$ as a child of $B_{j^*}$).

We claim that the inductive properties are satisfied. First remark that by construction $\overline{key}(a_{n+1})$ is distinct from all the keys in $K_n$; thus $B_0, \dots B_{n+1}$ are distinct blocks containing no element of $K$. Then note that $A_{n+1} \subseteq A_n \cup \{a_{n+1}\}$ and therefore $\mathcal{A}_{n+1} \subseteq \mathcal{A}_n \cup \{B_{n+1}\}$; as a consequence $\mathcal{F}_{n+1} \subseteq \mathcal{F}_n \cup \{B_{j^*}\}$. We now prove $C_{n+1}$.

$C_{n+1}$**.(a).** $\mathcal{T}_{n+1}$ is a binary tree, in fact $B_{j^*}$ has at most one child in $\mathcal{T}_n$. This follows from the fact that $D \models q\{a_{j^*}c\}$ with $\overline{key}(c)$ not occurring in $K_n$, so by Lemma 7 there can be at most one $B_i$, $i \leq n$, such that $D \models q\{a_{j^*}b_i\}$. Further for the only new parent-child pair $B_{j^*} = s_{n+1}(B_{n+1})$ we have $D \models q\{a_{j^*}b_{n+1}\}$.

$C_{n+1}$**.(b)** Let $B_l \in \mathcal{A}_{n+1} \cup \mathcal{F}_{n+1}$, $l > 0$, and let $B_i$ be a non-leaf descendant of $B_l$ in $\mathcal{T}_{n+1}$. Notice that since $B_l$ is not a leaf of $\mathcal{T}_{n+1}$, $B_l \in \mathcal{A}_n \cup \mathcal{F}_n$; moreover by construction of $\mathcal{T}_{n+1}$, $B_i$ is

either $B_{j^*}$ or a non-leaf of $\mathcal{T}_n$. In the latter case $C_n$.(b) implies $g(a_l) \subseteq key(a_i)$. In the case $B_i = B_{j^*}$, Claim 22 implies $g(a_l) \subseteq key(a_i)$.

$C_{n+1}$**.(c)** We only need to prove the property for $B_{j^*}$. By construction of $B_{n+1}$ it follows that either $\bar{g}(a_{j^*}) = \overline{key}(a_{j^*})$ or $\bar{g}(a_{j^*}) = \overline{key}(a_{n+1})$.

$C_{n+1}$**.(d)** $|A_{n+1} \cup K| \leq k$ follows from $C_{n+1}$. (a), $C_{n+1}$. (b) and $C_{n+1}$. (c) as follows. Consider the skeleton tree of $\mathcal{T}_{n+1}$ connecting the blocks in $\mathcal{N} := \mathcal{A}_{n+1} \cup \mathcal{F}_{n+1}$ *i.e.* the skeleton has nodes $\mathcal{N}$, and $B_i$ is the parent of $B_j$ in the skeleton iff $B_i$ is the least proper ancestor of $B_j$ belonging to $\mathcal{N}$ in $\mathcal{T}_{n+1}$. The skeleton has at most the rank of $\mathcal{T}_{n+1}$ (i.e. rank two by $C_{n+1}$.(a)) because the set $\mathcal{N}$ is closed under least common ancestor. Moreover the height of the skeleton tree is the maximum number of elements of $\mathcal{N}$ which can be found in a branch from root to leaf of $\mathcal{T}_{n+1}$ minus 1.

We now prove that this number is bounded. In fact given a branch from the root to a leaf of $\mathcal{T}_{n+1}$, let $\mathcal{I}$ be the (possibly empty) sequence of elements of $\mathcal{N}$ in the branch, excluding root and leaf. If this sequence is not empty let $B_l$ be its last element in the ancestor-descendant order. By $C_{n+1}$.(b) we have that for all $B_i \in \mathcal{I}$, $g(a_i) \subseteq key(a_l)$. This implies $|\{\bar{g}(a_i) \mid B_i \in \mathcal{I}\}| \leq \kappa$. We now prove that at most two different $B_i, B_j \in \mathcal{I}$ can have $\bar{g}(a_i) = \bar{g}(a_j)$.

Let $B_i$ and $B_j$ be such $\bar{g}(a_i) = \bar{g}(a_j)$. By $C_{n+1}$. By (c), there exists $i'$ and $j'$ such that $\bar{g}(a_i) = \overline{key}(a_{i'})$ and $\bar{g}(a_j) = \overline{key}(a_{j'})$. As all blocks are distinct, this implies that $i' = j'$. Moreover (c) implies that $B_i = B_{i'}$ or $B_{i'}$ is a child of $B_i$. This implies that $B_i$ is a child of $B_j$ or vice-versa.

It follows that $|\mathcal{I}| \leq 2\kappa$, and the number of elements of $\mathcal{N}$ in a branch of $\mathcal{T}_{n+1}$ is bounded by $2\kappa + 2$. Thus the skeleton of $\mathcal{T}_{n+1}$ is a binary tree of height bounded by $2\kappa + 1$. Moreover elements of $\mathcal{A}_{n+1}$ must have at most one child in the skeleton; thus $|\mathcal{A}_{n+1}| = |A_{n+1}| \leq 2^{2\kappa+1}$.

Since $A_{n+1} \cup K$ is the disjoint union of $A_{n+1}$ and $K \setminus \{a\}$, one has $|A_{n+1} \cup K| \leq 2^{2\kappa+1} + \kappa - 1 = k$.

Given this bound is now easy to see that $A_{n+1} \cup K \notin \Delta_k(q, D)$. In fact we have already observed that $A_{n+1} \subseteq A_n \cup \{a_{n+1}\}$ and that $A_n \cup K \cup \{a_{n+1}\}$ contains no $k$-sets.

$C_{n+1}$**.(e)** By construction $K_{n+1} = K_n \cup \{a_{n+1}\}$. Assume $D \models q\{ab\}$ for $a, b \in K_{n+1}$. By $C_n \cdot (e)$ this implies that $a = a_{n+1}$. So $b \notin A_n \cup K \cup \{a_{n+1}\}$ otherwise $\{a, b\}$ would be a $k$-set in $A_n \cup K \cup a_{n+1}$. Then $b = a_l$ for some $l \leq n$ with $a_l \notin A_n$, and $D \models q\{a_l, a_{n+1}\})$. Now recall that by construction $\overline{key}(a_{n+1})$ does not occur in $K_n$, so by definition of $A_n$ we must have $a_l \in A_n$, which is a contradiction.

This completes the proof of Lemma 21 and hence of Proposition 10. □

# E Sufficient conditions for the absence of tripath

In this section we give some sufficient syntactic conditions on the queries that imply that the queries that satisfy these conditions do not admit a TRIPATH. By Theorem 9 it follows that for these queries, CERTAIN($q$) is in PTIME.

The first case is when the query is non-branching: there is no fact that can be part in two solutions within a repair. For instance, the reader can verify that this is the case for the query $q_5 = R(\underline{x}\ yx)\ R(\underline{y}\ xu)$. For such queries clearly $q$ does not admit a TRIPATH because there can not be a center. Hence computing certain answers for non-branching queries is in PTIME.

We now consider a more involved syntactic condition. Let $q = AB$ be a 2way-determined query. We define $P_1$ as the most general atom such that there is an homomorphism from $\overline{key}(A)$ to $\overline{key}(P_1)$ and also from $B$ to $P_1$. Dually let $P_1'$ be the most general atom such that there is an homomorphism from $\overline{key}(B)$ to $\overline{key}(P_1')$ and also from $A$ to $P_1'$. Notice that $P_1 \sim P_1'$ and the idea

is to capture those blocks which can potentially have facts that match $A$ and also facts that can match $B$. We also define $P_0$ and $P_2$ are the atoms that match $A$ and $B$ when $P_1$ and $P_1'$ matches $B$ and $A$ respectively.

Formally, $P_0P_1$ is built as follows: For all $x, y \in vars(A) \cup vars(B)$, define $x \equiv y$ if there exists key positions $i, j$ such that $A(i) = A(j)$ and we have $B(i) = x$ and $B(j) = y$. For every equivalence class of variables fix a representative and let $h$ be the mapping that associates every variable $x$ the representative of the equivalence class to which $x$ belongs. We then let $h(AB) = P_0P_1$.

Now by construction, there is a homomorphism $h_1 : \overline{key}(A)$ to $\overline{key}(P_1)$. Consider the mapping $h_1'$ where for every variable $x \in vars(A) \cup vars(B)$ if $x \in key(A)$ then $h_1'(x) = h_1(x)$; otherwise $h_1'(x) = z$ where $z$ is a fresh variable.

Thus we have defined $P_0, P_1, P_1'$ and $P_2$ such that $P_1 \sim P_1'$ and there are homomorphsims from $AB$ to $P_0P_1$ and from $AB$ to $P_1'P_2$. For a given query $q$, we call the corresponding $P_0, P_1, P_1'$ and $P_2$ as its generalized 2-path. This name is justified by the following observation.

**Fact 24.** Let $q = AB$ be a 2way-determined query with $P_0, P_1, P_1'$ and $P_2$ as its generalized 2-path. Then for all database $D$ and for all facts $a, b, b', c \in D$ such that $b \sim b'$ and $D \models q(ab) \wedge q(b'c)$, there is a homomorphism from $(P_0P_1P_1'P_2)$ to $(abb'c)$.

Consider for instance the query $q_8 = R(\underline{xyzz}\ u)\ R(\underline{zzyu}\ x)$. The associated generalized 2-path is $P_0 = R(\underline{xyzz}\ y)$, $P_1 = R(\underline{zzyy}\ x)$, $P_1' = \overline{R(\underline{zzyy}\ x')}$ and $P_2 = R(\underline{yyzx'}\ z)$. The next syntactic condition that we define depends on the variables occurring in the key of $P_1$. Recall that $A[I]$ refers to the set of variables that occur in the positions $I$ in $A$.

**Lemma 25.** *Let $q = AB$ be a 2way-determined query and let $I, J$ be the subset of key positions such that $A[I] = B[J]$. If $key(P_1) = P_0[I]$ then $q$ does not admit a* TRIPATH.

Notice that the condition of the lemma is satisfied by $q_8$, so from Lemma 25 and Theorem 9, CERTAIN($q_8$) is in PTIME. To prove Lemma 25, we set up some definitions. Let $D$ be a database. A sequence of facts $\Pi = a_0b_0a_1b_1, \ldots a_n, b_n, a_{n+1}$ is called an alternating path if for all $i \leq n$ we have $a_i \sim b_i$ and $D \models q\{a_{i+1}b_i\}$, and if $j \neq i$ then $a_i \not\sim a_j$. An alternating patch is strict if $a_i \neq b_i$ for every $i$. Note that any path of a TRIPATH forms a strict alternating path.

*Proof.* Let $D$ be an arbitrary database. We will prove that there cannot be a TRIPATH $\Theta$ such that $\Theta \subseteq D$. First note that if $D$ does not contain any branching fact then clearly we cannot have any TRIPATH in $D$. So let $e$ be a branching fact in $D$ with $D \models q(de) \wedge q(ef)$. We will prove that there cannot exist a TRIPATH $\Theta \subseteq D$ with $def$ as the center facts of $\Theta$.

Note that there is a homomorphism from $(P_0P_1P_1'P_2)$ to $(deef)$. Also, by definition of $g(e)$, we always have $g(e) \subseteq key(e)$. Hence, if there is a TRIPATH $\Theta$ in $D$ with $def$ as the center facts then there exists an alternating path $\Pi = a_0b_0a_1b_1 \ldots a_m, b_m, a_{m+1}$ where $a_0 = d$, $n \geq 1$ and $g(e) \subseteq key(e) \not\subseteq key(a_{m+1})$ where $\Pi$ does not intersect the block of $e$.

To arrive at a contradiction, we show that for any strict alternating path $\Pi = a_0b_0a_1b_1 \ldots a_nb_na_{n+1}$ where $a_0 = d$ if $\Pi$ does not intersect with the block of $e$ then $key(e) \subseteq key(a_i)$ for all $i \leq n + 1$. This implies $g(e) \subseteq key(a_{n+1})$, thus contradicting the existence of the TRIPATH.

To prove the claim we verify the following properties for all $i \leq n$ by induction on $i$:

(a) If $i = 0$ then $key(e) = a_i[I]$.

(b) If $i > 0$ and $D \models q(b_ia_{i+1})$ then $key(e) \subseteq a_{i+1}[J]$

(c) If $i > 0$ and $D \models q(a_{i+1}b_i)$ then $key(e) \subseteq a_{i+1}[I]$

(d) If $i > 0$ and $D \models q(b_{i-1}a_i)$ then $D \not\models q(b_ia_{i+1})$

Recall that $I, J$ are subsets of key positions such that $A[I] = B[J]$. In the base case when $i = 0$ note that there is a homomorphism from $P_0P_1$ to $de$ and hence $key(e) = d[I] = a_0[I]$.

Now assume that the claims hold upto $i - 1 \leq n$ and we will verify them for $i$. Recall that $D \models q\{b_{i-1}a_i\} \wedge q\{b_ia_{i+1}\}$. We prove the invariants by considering all possible cases of these solutions.

- If $D \models q(b_{i-1}a_i) \wedge q(a_{i+1}b_i)$ (in this case Item (c) is to be verified). By induction hypothesis, $key(e) \subseteq a_i[J]$ and also $b_i[J] = a_i[J]$. Now since there is a homomorphism from $AB$ to $a_{i+1}b_i$ we have $a_{i+1}[I] = b_i[J]$. Hence $key(e) \subseteq a_{i+1}[I]$.

- The case $D \models q(a_ib_{i-1}) \wedge q(b_ia_{i+1})$ is symmetric to the previous case, where Item (b) can be verified.

- If $D \models q(a_ib_{i-1}) \wedge q(a_{i+1}b_i)$ (in this case Item (c) is to be verified). By induction hypothesis, $key(e) \subseteq a_i[I]$. Note that in this case, there is a homomorphism from $P_0P_1P_1'P_2$ to $(a_{i+1}b_ia_ib_{i-1})$. Note that $P_1[I] \subseteq key(P_1) = P_0[I] = P_1[J]$ and hence $b_i[I] \subseteq b_i[J]$. Since $a_i[I] = b_i[I]$ and $a_i[J] = b_i[J]$ and by induction $key(e) \subseteq a_i[I]$, it follows that $key(e) \subseteq b_i[J]$.

  Now since there is a homomorphism from $AB$ to $a_{i+1}b_i$ we have $a_{i+1}[I] = b_i[J]$. Hence $key(e) \subseteq a_{i+1}[I]$.

- If $D \models q(b_{i-1}a_i) \wedge q(b_ia_{i+1})$ (in this case we get a contradiction thus verifying Item (d)). Suppose this happens then note that there is a homomorphism from $\overline{key}(P_1)$ to $\overline{key}(a_i)$. Towards contradiction, we will prove that $a_i \sim e$ or $a_i \sim a_j$ for some $j < i$.

  Let $j < i$ be the largest index such that $D \models q(a_jb_{j-1}) \wedge q(a_{j+1}b_j)$. If no such $j$ exists, let $j = 0$. Note that by assumption $j < i$.

  By maximality of $j$ and because of Item (d), for all $k$ such that $j+2 \leq k \leq i$ if $D \models q(b_{k-1}a_k)$ then $D \models q(a_{k-1}b_{k-2})$ and if $D \models q(a_kb_{k-1})$ then $D \models q(b_{k-2}a_{k-1})$. If $j > 0$, a simple inductive argument shows that this implies $a_i[J] = a_j[J]$ and we set $u = a_j$. If $j = 0$ the same argument implies $e[J] = a_i[J]$ and we set $u = e$.

  Thus, in both cases $u[J] = a_i[J]$ and there is a homomorphism from $\overline{key}(P_1)$ to both $\overline{key}(a_i)$ and $\overline{key}(u)$. We also have $key(P_1) = P_1[J]$ and combining this with $a_i[J] = u[J]$, we get $a_i \sim u$, the desired contradiction. □

There is also another way to characterize the above syntactic condition.

**Lemma 26.** *Let $q = AB$ be a 2way-determined query and let $I, J$ be the sub-set of key positions such that $A[I] = B[J]$. Then: $key(P_1) = P_0[I]$ iff $key(P_1) \subseteq key(P_0)$*

*Proof.* If $key(P_1) = P_0[I]$ then the claim follows since $P_0[I] \subseteq key(P_0)$. For the converse, assume $key(P_1) \subseteq key(P_0)$. We verify that $key(P_1) = P_0[I]$. Pick any key position $i$. If $i \in J$ then clearly $P_1[i] \in P_0[J]$.

Assume $i \notin J$. Note that $key(P_1) \subseteq key(P_0)$. By construction of $P_0P_1$ it implies that there is some position $i'$ such that $B[i] \equiv A[i']$. But this is possible only if there exists $j, j'$ such that $B[i] = B[j]$ and $A[i'] = B[j']$ and $A[j] = A[j']$. Hence, $i' \in I$ and $j' \in J$ and $P_1[i] = P_1[j'] = P_0[i']$. Thus, $P_1[i] \in P_0[I]$.

□

Symmetrically, if we assume that $key(P_1') \subseteq key(P_2)$ then also we can prove an analogous result as Lemma 25 and Lemma 26. Thus, we have the following corollary.

**Corollary 27.** *Let $q = AB$ be a 2way-determined query with $P_0, P_1, P_1'$ and $P_2$ as the generalized 2-path. If $key(P_1) \subseteq key(P_0)$ or $key(P_1') \subseteq key(P_2)$ then $q$ does not admit a* TRIPATH *(and* CERTAIN$(q)$ *is in* PTIME*).*

# F Proofs for Section 9 (Fork-tripath and coNP-hardness)

**Lemma 13.** *Let $\phi$ be a 3-sat formula where every variable occurs at most three times. $\phi$ is satisfiable iff $D[\phi] \not\models$* CERTAIN$(q)$.

Before proving the Lemma, note that the following proposition holds for nice fork-TRIPATH.

**Proposition 28.** *Let $q$ be a 2way-determined query and $\Theta$ be a nice fork-*TRIPATH *of $q$ where $U_e$ is the root block and $U_d, U_f$ are the leaf blocks with $u_d, u_e$ and $u_f$ as the unique facts in $U_d, U_e$ and $U_f$ respectively. Let $D \supseteq \Theta$ such that for every internal block $B$ of $\Theta$, $B$ is also a block of $D$ (i.e. $D$ does not contain any extra facts in $B$). Then:*

1. *For every repair $r$ of $D$ if $u_e \in r$ and $r \models \neg q$ then both $u_d, u_f \notin r$.*
2. *For every repair $r$ of $\Theta$ if $r$ contains $a(B)$ for every block $B$ in $\Theta$ then $r \not\models q$.*
3. *For every repair $r$ of $\Theta$ if $r$ only contains $b(B)$ for every block $B$ in $\Theta$ then $r \not\models q$.*

*Proof of lemma 13.* Assume first that $\phi$ is satisfiable. Let $h$ be an assignment of the variables of $\phi$ that makes the query true. For each clause $C$ of $\phi$ we set $h(C)$ to be a literal that makes the clause true as specified by $h$. We define the falsifying repair denoted by $r[h]$ as follows:

Let $l$ be a variable of $\phi \cap V_3$, such that $l$ (or $\neg l$) occur once positively - let $C[l]$ be this clause - and twice negatively - let $C_1[l]$, $C_2[l]$ be the two corresponding clauses.

If $l = h(C)$ (then $\neg l \neq h(C_1)$ and $\neg l \neq h(C_2)$ as $h$ is a satisfying assignment) then in each non-leaf block of $B$ of $\Theta_{l,C}$ we select the fact $a(B)$ and in each non-head block $B$ of $\Theta_{l,C_1} \cup \Theta_{l,C_2}$ the fact $b(B)$. After this, if there is a block $B$ in $\Theta_{l,C}$ for which a fact has not been selected, it is because $B$ contains a single fact $\Theta_{l_C} \cup \Theta_{l,C_1} \cup \Theta_{l,C_2}$. By construction we have added a fresh fact $c$ to $B$ in $D[\phi]$ which does not form a solution with any other fact. Select $c$ for the block $B$.

If $\neg l = h(C_1)$ or $\neg l = h(C_2)$, (then $l \neq h(C)$ as $h$ is a satisfying assignment) then we select in each non-head block $B$ of $\Theta_{l,C}$ the fact $b(B)$ and in each non-leaf block $B$ of $\Theta_{l,C_1} \cup \Theta_{l,C_2}$ we select the fact $a(B)$. (In this case we have selected one fact for all blocks of $\Theta(l, C) \cup \Theta_{l,C_1} \cup \Theta_{l,C_2}$ )

Otherwise we select select in each non-head block $B$ of $\Theta_{l,C}$ the fact $b(B)$ and in each non-leaf and non-head block $B$ of $\Theta_{l,C_1} \cup \Theta_{l,C_2}$ the fact $a(B)$. (In this case also we have selected one fact for all blocks of $\Theta(l, C) \cup \Theta_{l,C_1} \cup \Theta_{l,C_2}$ )

The construction is similar if $l \in V_2$. The reader can verify that $r[h]$ can not make the query true because within every $\Theta_{l,C}$ since we only have $a(B)$ facts or only $b(B)$ facts (c.f proposition 28 (2)) and there are no solutions that involve facts across two distinct copies of TRIPATH in $D[\phi]$.

For the converse let $r$ be a repair of $D[\phi]$ such that $r \models \neg q$. Using $r$ we construct a satisfying assignment $h$ as follows. For each clause $C$ of $\phi$ consider the block of $C$. Note that $r$ has selected one fact from $C$. By construction this fact corresponds to a literal $l$ of $\phi$. We set $h(l)$ to true. In order to show that $h$ is a satisfying assignment it suffices to show that if $l$ is selected by $r$ in a clause $C$ then $\neg l$ can not be selected by $r$ in a clause $C'$. This is a consequence of proposition 28 (1): if $r$ selects the heads of both $\Theta_{l,C}$ and $\Theta_{l,C'}$, their common leaf has no representative in $r$, a contradiction. □

# G Sufficient conditions for the presence of a fork-tripath

To identify some sufficient conditions where a query admits a fork-TRIPATH, we first set up some definitions. Let $q = AB$ where $q$ is 2way-determined. Recall that if $D \models q(ab)$ then there is an homomorphism from $AB$ to $ab$.

We construct three atoms $F_0F_1F_2$ such that whenever there are facts $a, b, c$ such that $D \models q(ab) \wedge q(bc)$ (*i.e.* whenever $abc$ is a fork) then there is an homomorphism mapping $F_0F_1F_2$ to $abc$.

Formally, $F_0$ and $F_1$ are built as follows: For all $x, y \in vars(A) \cup vars(B)$, define $x \equiv y$ if there exists positions $i, j$ such that $A(i) = A(j)$ and we have $B(i) = x$ and $B(j) = y$. For every equivalence class of variables fix a representative and let $h$ be the mapping associating every variable $x \in vars(A) \cup vars(B)$ to the representative of the equivalence class to which $x$ belongs. Define $F_0F_1$ be $h(AB)$. By construction there is an homomorphism from $AB$ to $F_0F_1$. To define $F_2$, consider the following mapping $h'$ where for every variable $x \in vars(A) \cup vars(B)$: if $x \in vars(A)$ then let $i$ be a position where $x$ occurs in $A$ and let $y$ be the variable occurring at position $i$ in $B$, then $h'(x) = h(y)$; otherwise if $x$ does not occur in $A$ then let $h'(x)$ be a fresh

new variable. Notice that by construction of $h$, the definition of $h'$ does not depend on the choice of $i$ and therefore $h'$ is well defined. Notice also that $F_1 = h'(A)$. Define $F_2 = h'(B)$.

The triple $F_0F_1F_2$ is called the fork of $q$. Note that the triple $F_0F_1F_2$ depends on the ordering of the atoms $A$ and $B$ in $q$. It also has the desired property:

**Proposition 29.** *Let $F_0F_1F_2$ be the fork of $q = AB$. Then for all database $D$ and for all facts $a, b, c$ if $D \models q(ab) \wedge q(bc)$ then there is a homomorphism from $F_0F_1F_2$ to $abc$.*

**Example 30.** We now illustrate the different forms that the fork can take (note that all these queries are 2way-determined)

- Consider the query $R(\underline{x}\ yx)\ R(\underline{y}\ xu)$. Then $F_0F_1F_2$ is $R(\underline{x}\ yx)\ R(\underline{y}\ xy)\ R(\underline{x}\ yv)$. So we have $F_2 \sim F_0$ and hence in any repair every fact will be a part of at most 1 solution, i.e. there is no branching fact. Thus, for this query the fork does not play any important role.
- Consider now the query $R(\underline{x}\ yu)\ R(\underline{y}\ yx)$. In this case $F_0F_1F_2$ is $R(\underline{x}\ yu)\ R(\underline{y}\ yx)\ R(\underline{y}\ yy)$. Here we have $F_2 \sim F_1$ and hence in this case also every fact in a repair will be a part of at most 1 solution.
- In the query $R(\underline{x}\ yz)\ R(\underline{y}\ zx)$, the fork $F_0F_1F_2$ is $R(\underline{x}\ yz)\ R(\underline{y}\ zx)\ R(\underline{z}\ xy)$. In this case there is also a homomorphism from $AB$ to $F_2F_0$. Hence in all database $D$ and for all facts $a, b, c \in D$ if $D \models q(ab) \wedge q(bc)$ then $D \models q(ca)$. Such queries can be verified to be clique-queries and hence CERTAIN($q$) can be computed in polynomial time (cf. Theorem 17).
- Finally, consider the query $R(\underline{x}\ yz)\ R(\underline{y}\ ux)$ then $F_0F_1F_2$ is $R(\underline{x}\ yz)\ R(\underline{y}\ ux)\ R(\underline{u}\ vy)$. In this case $F_0F_1F_2$ have pair-wise distinct keys and also there is no homomorphism from $AB$ to $F_2F_1$. This is the most general case.

Intuitively a query $q$ can admit a fork-TRIPATH only if there is some fork (*i.e.* there exists a database $D$ and facts $d, e, f \in D$ such that $d, e, f$ have mutually distinct keys and $q(de) \wedge q(ef)$ but not $q(fd)$). This is possible iff $F_0, F_1$ and $F_2$ have mutually distinct keys and there is no homomorphism from $AB$ to $F_2F_0$. A 2way-determined query $q$ is called a **fork query** if the corresponding fork atoms $F_0, F_1$ and $F_2$ have mutually distinct keys and there is no homomorphism from $AB$ to $F_2F_0$.

## G.1 Syntactic conditions for a query to admit a fork-tripath

We exhibit a syntactic condition implying that a query admits a fork-TRIPATH. The condition is about the key (non-)inclusion conditions with respect to $F_0, F_1$ and $F_2$.

**Lemma 31.** *Let $q$ be a 2way-determined fork query and $F_0F_1F_2$ be the fork of $q$. If $key(F_0) \not\subseteq key(F_1)$, $key(F_1) \not\subseteq key(F_0)$, $key(F_2) \not\subseteq key(F_1)$ and $key(F_1) \not\subseteq key(F_2)$ then $q$ admits a fork-*TRIPATH.

For instance the query $q_9 = R(\underline{xvy}\ zuv)\ R(\underline{vuz}\ xyz)$. Note that $q_9$ is 2way-determined and the corresponding fork is given by $\overline{F_0}\overline{F_1}F_2 = R(\underline{xvy}\ uuv)\ R(\underline{vuu}\ xyu)\ R(\underline{uyx}\ vux)$ which satisfies the conditions in the lemma and hence admits a fork-TRIPATH (so by theorem 12 CERTAIN($q_9$) is coNP-hard).

*Proof of lemma 31.* In this case we will directly build a fork-TRIPATH for $q$. Let $def$ be the three facts such that there is an isomorphism from $F_0F_1F_2$ to $def$. By the assumptions, $def$ have pairwise distinct keys and do not form a triangle and $def$ will be the center of the TRIPATH that we construct. Notice that in this case $g(e) = key(e)$. So the root and the leaves of the TRIPATH should exclude a variable from $key(e)$.

Because $key(d) \not\subseteq key(e)$ and $key(e) \not\subseteq key(f)$, we can take as leaves of the TRIPATH the two blocks containing a single fact, one containing the fact $d$ and one containing the fact $f$. It remains to construct the root block and the path from the root block to a block containing the branching fact $e$. This is done as follows:

Let $x \in key(F_0) \setminus key(F_1)$ and $y \in key(F_1) \setminus key(F_0)$ and let $z \in key(F_2) \setminus key(F_1)$. Note that we have $x, z \in vars(F_1) \setminus key(F_1)$ (the inclusion into $vars(F_1)$ is because $q$ is 2way-determined) and $y \in vars(F_0) \setminus key(F_1)$.

Let $h$ be the homomorphism from $AB$ to $de$. Construct $h_1$ using $h$ where for every variable $w$ if $w \neq x, z$ then $h_1(w) = h(w)$ and $h_1(x), h_1(z)$ are two fresh domain elements. Let $h_1(A'B') = a_1e'$. Since $x, z \notin key(e)$, we have $e' \sim e$. Since $x \in key(A')$ we have $a_1$ in a fresh block $B_a$. Also since $y \notin key(A')$ we have $key(e) \not\subseteq key(a_1)$.

Thus, the TRIPATH $\Theta$ has the blocks $\{B_a, B_d, B_e, B_f\}$ rooted at $B_a$ with leaf blocks $B_d = \{d\}$ and $B_f = \{f\}$ and branching block $B_e = \{e, e'\}$. $B_e$ is the successor of $B_a$ and both $B_d, B_f$ are the successor blocks of $B_e$. We also have $a(B_a) = a$; $b(B_e) = e', a(B_e) = e$; $b(B_d) = d$ and $b(B_f) = f$.

□

The previous syntactic condition can be turned into a semantic condition. A query is said to have to have **uniform triangles** if for all database $D$ and all triangle $def$ in $D$ we have $key(e) = key(d) = key(f)$. The following result is a simple consequence of Lemma 31 and Theorem 12.

**Corollary 32.** *Let $q$ be a 2way-determined fork query. If $q$ does not have uniform triangles then $q$ admits a fork-*TRIPATH*, and thus* CERTAIN$(q)$ *is* coNP*-hard.*

*Proof.* Consider the fork $F_0F_1F_2$ of $q$. We show that the variable non-inclusion conditions of Lemma 31 hold. In fact assume by contradiction that at least one of those inclusions hold; then note that, in any database, if $abc$ is a triangle, there is a homomorphism from $F_0F_1F_2$ to each of $(abc)$ $(bca)$ and $(cab)$. Hence we will have $key(a) = key(b) = key(c)$. This implies that $q$ has uniform triangles which is a contradiction. Hence by Lemma 31, $q$ admits a fork-TRIPATH, and thus CERTAIN$(q)$ is coNP-hard, by Theorem 12. □

## G.2 Triangle-fork connections

In this section we give a second semantic condition implying the existence of a fork-TRIPATH. We start by giving some new definitions. Two blocks $B, B'$ of a database $D$ are said to be **$q$-connected** if $(B, B')$ belongs to the reflexive symmetric transitive closure of $\{(B_1, B_2) \mid \exists a \in B_1, b \in B_2$ such that $D \models q\{ab\}\}$. Note that this is an equivalence relation among blocks of a database. A database $D$ is $q$-connected if every pair of blocks of $D$ is $q$-connected.

we now extend this notion to facts : two facts $c, d$ in a database $D$ are said to be $q$-connected if the block of $c$ and the block of $d$ are $q$-connected in $D$. Note that facts $c, d$ in $D$ are $q$-connected iff $c \sim d$ or there is an alternating path $\Pi = a_0b_0 \dots a_nb_na_{n+1}$ in $D$ such that $c \sim a_0$ and $d \sim a_{n+1}$. Note also that $D$ is $q$-connected iff all pairs of facts of $D$ are $q$-connected.

We say that a query $q$ admits a triangle-fork $q$-connected database if there exists a $q$-connected database $D$ and there exists $\Theta \subseteq D$ where $\Theta$ is a triangle-TRIPATH and $D$ also contains a fork.

For instance consider the query $q_{10} = R(\underline{xy_1z_1}\ z_2y_2)\ R(\underline{z_1xy_2}\ y_1x)$. For this query there is a triangle-fork $q$-connected database $D$ such that $\Theta \subseteq D$ is a triangle-TRIPATH and $D$ also contains a fork. From the next theorem it follows that CERTAIN$(q_{10})$ is coNP-hard.

**Theorem 33.** *Let $q$ be a 2way-determined query. If $q$ admits a triangle-fork $q$-connected database, then $q$ also admits a fork-*TRIPATH*.*

The rest of this section is devoted to the proof of Theorem 33. We proceed in two steps. In the first step we define some syntactic conditions and prove that any query satisfying these conditions admits a fork-TRIPATH (cf. Proposition 35). In the second step we show that if a query $q$ admits a triangle-fork $q$-connected database then the syntactic conditions are satisfied and hence $q$ admits a fork-TRIPATH (cf. Proposition 36 and Proposition 46).

Recall that an alternating path $\Pi = e_0f_0e_1f_1 \dots e_n$ is *strict* if $e_i \neq f_i$ for all $i < n$. Note that any branch in a TRIPATH forms a strict alternating path. We say that a query $q$ admits a one-sided fork-TRIPATH if there exists a database $D$, a fork $def$ in $D$ and a strict alternating path $\Pi$ from some $a_0 \in \{d, e, f\}$ to some $b$ such that (1) $g(e) \not\subseteq key(b)$ and (2) $\Pi$ does not intersect the

blocks of the other two facts from the fork $def$ which are different from $a_0$. Informally, one-sided fork-TRIPATH has only one branch out of three for the full TRIPATH.

Let $\mathsf{q\text{-}fix}$ be the set of pairs of key positions of $q$ computed by the following fix-point algorithm:

$$\begin{aligned}
\mathsf{q\text{-}fix}_0 =& \{(i,j) \mid A[i] = A[j]\} \cup \{(i,j) \mid B[i] = B[j]\} \\
\mathsf{q\text{-}fix}_{n+1} =& \Big( \quad \mathsf{q\text{-}fix}_n \\
& \cup \{(i,j) \mid \exists\ (i',j') \in \mathsf{q\text{-}fix}^n \text{ s.t } A[i] = B[i'] \text{ and } A[j] = B[j']\} \\
& \cup \{(i,j) \mid \exists\ (i',j') \in \mathsf{q\text{-}fix}^n \text{ s.t } B[i] = A[i'] \text{ and } B[j] = A[j']\} \ \Big)^* \\
\mathsf{q\text{-}fix} =& \bigcup_n \mathsf{q\text{-}fix}_n
\end{aligned}$$

Note that $\mathsf{q\text{-}fix}$ is an equivalence relation over key positions. The following lemma is an important property of $\mathsf{q\text{-}fix}$.

**Lemma 34.** *Let $q$ be a 2way-determined query. Let $D$ be a database and $def$ be a triangle in $D$. Then for all $(i,j) \in \mathsf{q\text{-}fix}$ and $a \in \{d,e,f\}$ we have $a[i] = a[j]$.*

The following proposition provides a sufficient condition for extending one-sided fork-TRIPATH to a full TRIPATH.

**Proposition 35.** *Let $q = AB$ be a query that is 2way-determined. Then $q$ admits a fork-*TRIPATH *if all of the following conditions hold:*

1. *There exists $x \in key(A)$ such that for all key position $j$ if $A[j] = x$ then for all $(j,k) \in \mathsf{q\text{-}fix}$ we have $A[k] \notin key(B)$*
2. *There exists $z \in key(B)$ such that for all key position $j$ if $B[j] = z$ then for all $(j,k) \in \mathsf{q\text{-}fix}$ we have $B[k] \notin key(A)$*
3. *$q$ admits one-sided fork-*TRIPATH

*Proof.* Let $x$ and $z$ be two variables given by (1) and (2). Let $V_x = \{x' \mid$ there exists $(j,k) \in \mathsf{q\text{-}fix}$ where $A[j] = x$ and $A[k] = x'\}$ and let $V_z = \{z' \mid$ there exists $(j,k) \in \mathsf{q\text{-}fix}$ where $B[j] = z$ and $B[k] = z'\}$. From (1), $V_x \cap key(B) = \emptyset$ and $V_x \cap key(A) \neq \emptyset$. Similarly from (2), $V_z \cap key(A) = \emptyset$ and $V_z \cap key(B) \neq \emptyset$.

From (3) we get a database $D$ with a fork $def$ and a fact $a_0 \in \{d,e,f\}$ such that there is an alternating path $\Pi$ from $a_0$ to $b$ such that $g(e) \not\subseteq key(b)$ and $\Pi$ does not intersect the blocks of the other two facts from $def$ which are different from $a_0$. Assume that $a_0 = d$ (the other cases are treated analogously).

So we have an alternating path $\Pi = a_0 b_0 a_1 b_1, \ldots a_n$ where $a_0 = d$ such that $g(e) \not\subseteq key(a_n)$. We now construct an alternating path $\Pi' = ee' c_0 d_0 c_1 d_1, \ldots c_n$ starting at $e$ such that $g(e) \not\subseteq key(c_n)$. We construct $\Pi'$ by induction where for all $i \leq n$ we maintain the following invariants:

(a) For every key position $j$ if $c_i[j] \in key(d) \cup key(e) \cup key(f)$ then $a_i[j] = c_i[j]$.

(b) For every key position $j$, if $c_i[j] \neq a_i[j]$ then for all key positions $k$ if $(j,k) \in \mathsf{q\text{-}fix}$ then $c_i[j] = c_i[k]$

(c) If there is a homomorphism from $\overline{key}(A)$ to $\overline{key}(a_i)$ then there is a homomorphism from $\overline{key}(A)$ to $\overline{key}(c_i)$.

(d) If there is a homomorphism from $\overline{key}(B)$ to $\overline{key}(a_i)$ then there is a homomorphism from $\overline{key}(B)$ to $\overline{key}(c_i)$.

Note that from item (a) and the fact that in $\Pi$ we have $g(e) \not\subseteq key(a_n)$, it follows $g(e) \not\subseteq key(c_n)$ as desired.

In the base case of the induction, let $h$ be the homomorphism from $AB$ to $de$. Let $\alpha$ be a fresh domain element. Define $h'$ constructed from $h$ as follows. Let $y$ be a variable of $q$. If $y \in V_x$ then set $h'(y) = \alpha$; otherwise set $h'(y) = h(y)$. Let $c_0e'$ be $h'(AB)$. Note that $h$ and $h'$ agree on all variables in $key(B)$ and hence $e \sim e'$; moreover $e \neq e'$ since $\alpha$ occurs in $e'$ but not in $e$. Also we have $\alpha \in key(c_0)$ and hence $c_0$ is in a fresh block. Now we verify the invariants.

(a) Holds by construction.

(b) Pick a position $j$ such that $c_0[j] \neq a_0[j]$. By construction $c_0[j] = \alpha$ and by definition, for all key positions $k$ if $(j,k) \in \mathsf{q\text{-}fix}$ then $c_0[k] = \alpha$ as desired.

(c) Trivially holds by construction.

(d) Assume that there is a homomorphism from $\overline{key}(B)$ to $\overline{key}(a_0)$. Suppose $j,k$ are two positions such that $B[j] = B[k]$ then we have $a_0[j] = a_0[k]$. Also note that $(j,k) \in \mathsf{q\text{-}fix}$ and hence either $c_0[j] = c_0[k] = \alpha$ or $c_0[j] = a_0[j] = a_0[k] = c_0[k]$.

For the induction step, assume that we have the sequence $ee'c_0d_0 \ldots c_i$ for some $i < n$. Let $\alpha$ be a fresh domain element. Since $\Pi$ is an alternating path, we have either $q(a_{i+1}b_i)$ or $q(b_ia_{i+1})$. We define $d_i$ and $c_{i+1}$ depending on these two cases.

- If there is a homomorphism $h$ from $AB$ to $a_{i+1}b_i$ then, as $a_i \sim b_i$, there is a homomorphism from $\overline{key}(B)$ to $\overline{key}(a_i)$. This implies by Item (d) that there is a homomorphism $h_1$ from $\overline{key}(B)$ to $\overline{key}(c_i)$. Construct $h'$ as follows. Let $y$ be a variable of $q$. If $y \in key(B)$ then set $h'(y) = h_1(y)$. If $y \in V_x$ then set $h'(y) = \alpha$. If there are key positions $k,l$ such that $(k,l) \in \mathsf{q\text{-}fix}$, $A[k] = y$, $A[l] = B[l']$ for some key position $l'$ and $c_i[l'] \neq a_i[l']$ then set $h'(y) = c_i[l']$; otherwise set $h'(y) = h(y)$. Let $c_{i+1}d_i = h'(AB)$.

  We first argue that $h'$ is well defined, i.e. $h'(y)$ does not depend on the choice of $k,l$ in the third case. To see this, assume that there exists $(k_1,l_1) \in \mathsf{q\text{-}fix}$ such that $A[k_1] = y$ where $A[l_1] = B[l'_1]$ for some key position $l'_1$ and $c_i[l'_1] \neq a_i[l'_1]$. Then we should have $h'(y) = c_i[l'_1]$. Now assume that there exists another $(k_2,l_2) \in \mathsf{q\text{-}fix}$ such that $A[k_2] = y$ where $A[l_2] = B[l'_2]$ for some key position $l'_2$. To prove that $h'$ is well defined, it is sufficient to show that $c_i[l'_1] = c_i[l'_2]$.

  Note that since $A[k_1] = A[k_2] = y$ we have $(k_1,k_2) \in \mathsf{q\text{-}fix}$. By transitivity and symmetry, we have $(l_1,l_2) \in \mathsf{q\text{-}fix}$. But as $B[l'_1] = A[l_1]$ and $B[l'_2] = A[l_2]$ we then get $(l'_1,l'_2) \in \mathsf{q\text{-}fix}$. Further, since $c_i[l'_1] \neq a_i[l'_1]$ by Item (b) we obtain $c_i[l'_1] = c_i[l'_2]$.

- If there is a homomorphism $g$ from $AB$ to $b_ia_{i+1}$ then, as $a_i \sim b_i$, there is a homomorphism from $\overline{key}(A)$ to $\overline{key}(a_i)$. This implies by Item (c) there is a homomorphism $g_1$ from $\overline{key}(A)$ to $\overline{key}(c_i)$. The construction of $g'$ is then done in the same wasy as $h'$ above, replacing $V_x$ by $V_z$. We set $d_ic_{i+1}$ as $g'(AB)$.

Note that in both cases of the definition of $d_i$ and $c_{i+1}$, we have $d_i \sim c_i$ with $d_i \neq c_i$ and $\alpha \in key(c_{i+1})$. Hence $c_{i+1}$ is in a fresh block. We now verify the invariant. We only do the case when $c_{i+1}d_i = h'(AB)$. The case when $d_ic_{i+1} = g'(AB)$ is done similarly by symmetry.

(a) Let $j$ be a position such that $c_{i+1}[j] \in key(d) \cup key(e) \cup key(f)$. Let $y = A[j]$. We have $c_{i+1}[j] = h'(y)$.

  If $y = B[j']$ for some key position $j'$ then $c_i[j'] = d_i[j'] = c_{i+1}[j]$. It follows that $c_i[j'] \in key(d) \cup key(e) \cup key(f)$ and by induction that $a_i[j'] = c_i[j']$. This implies $b_i[j'] = d_i[j']$ and hence $a_{i+1}[j] = c_{i+1}[j]$.

  Clearly $y$ can not be in $V_x$ as $\alpha$ is a fresh new value.

Assume there exists key positions $k, l$ such that $(k, l) \in \mathsf{q\text{-}fix}$, $A[k] = y$, $A[l] = B[l']$ for some key position $l'$ and $c_i[l'] \neq a_i[l']$. In this case $c_{i+1}[j] = c_i[l']$. This implies $c_i[l'] \in key(d) \cup key(e) \cup key(f)$ and by induction hypothesis $a_i[l'] = c_i[l']$ which is a contradiction to the assumption. So this case does not apply.

In the remaining case, by definition we have $c_{i+1}[j] = h(y) = a_{i+1}[j]$ as desired.

(b) Let $j$ be some key position such that $c_{i+1}[j] \neq a_{i+1}[j]$. Let $y = A[j]$. Recall that $c_{i+1}[j] = h'(y)$. We do a case analysis depending on how $h(y)$ is defined.

- If $y = B[j']$ for some key position $j'$. Then $c_{i+1}[j] = c_i[j']$. Notice that $c_i[j'] \neq a_i[j'] = a_{i+1}[j]$.
  Now pick any position $k$ such that $(j, k) \in \mathsf{q\text{-}fix}$. By definition of $V_x$, this implies that $A[k] \notin V_x$.
  Assume first $A[k] = B[k']$ for some key position $k'$. By definition $(j', k') \in \mathsf{q\text{-}fix}$. From Item (b) by induction for $j'$ we have $c_i[k'] = c_i[j']$. Hence $c_{i+1}[j] = c_{i+1}[k]$.
- If $y \in V_x$ then by definition for all key position $k$ such that $(j, k) \in \mathsf{q\text{-}fix}$ we have $c_{i+1}[j] = c_{i+1}[k] = \alpha$.
- Assume now there are key positions $(k, l) \in \mathsf{q\text{-}fix}$ such that $A[k] = y$, and $A[l] = B[l']$ for some key position $l'$ and $c_i[l'] \neq a_i[l']$. In this case $h(y) = c_i[l']$ and $c_i[l'] \neq a_i[l']$. Now pick any key position $k'$ such that $(j, k') \in \mathsf{q\text{-}fix}$. Set $k''$ such that $A[k'] = B[k'']$.
  If $k''$ is a key position then by definition $(k'', l') \in \mathsf{q\text{-}fix}$ and since $c_i[l'] \neq a_i[l']$, from item (b) by induction we have $c_i[l'] = c_i[k'']$. Hence $c_{i+1}[k'] = d_i[k''] = c_i[k''] = c_i[l'] = c_{i+1}[j]$.
  If $k''$ is a non-key position then by transitivity $(k', l) \in \mathsf{q\text{-}fix}$ which implies $(k, l) \in \mathsf{q\text{-}fix}$. By definition of $h'_1$, $h'_1(A[k']) = c_i[l']$ because we are in the second case as witnessed by the fact that $(k', l)$ is in $\mathsf{q\text{-}fix}$. Hence $c_{i+1}[k'] = c_i[l']$. Now $c_{i+1}[j] = c_i[l'] = c_i[k'] = c_{i+1}[k]$ as desired.
- In the last case we have $c_{i+1}[j] = a_{i+1}[j]$, so this case does not apply.

Similarly if $c_{i+1} = g'_1(B)$ then we can argue that the claim holds.

(c) This is immediate as there is a homomorphism from $\overline{key}(A)$ to $\overline{key}(c_{i+1})$.

(d) Assume there is a homomorphism from $\overline{key}(B)$ to $\overline{key}(a_{i+1})$.Let $j, k$ are two key positions such that $B[j] = B[k]$. We need to show that $c_{i+1}[j] = c_{i+1}[k]$. Notice that we have $a_{i+1}[j] = a_{i+1}[k]$. By definition $(j, k) \in \mathsf{q\text{-}fix}$. The result is then a simple consequence of Item (b) as either all elements in a $\mathsf{q\text{-}fix}$-class are equal or they are equal to the same position in $a_{i+1}$.

Now using the alternating path $ee'c_0d_0 \dots c_n$, we can build the alternating path $ff'e_1e'_1c'_0d'_0 \dots c'_n$ exactly as we did above to construct $ee'c_0d_0 \dots c_n$ from $a_0b_0 \dots a_n$. Let $\Theta$ be the result construction. We claim that $\Theta$ is the required TRIPATH. By construction the center is a fork and each blocks are pairwise distinct as their key contains a fresh new element. Also by construction an element from $g(e)$ is excluded in the root and in each of the leaves. It remains to verify that for all facts $t \in \Theta$ if $\Theta \models q\{et\}$ then $t \in \{d, f\}$. Note that as $q$ is 2way-determined, if $\Theta \models q\{et\}$ then $t \sim d$ or $t \sim f$. By hypothesis on $\Pi$, if $t \sim d$ then $t = d$. Consider the variable $x$. As $q$ is 2way-determined, $x \in B$. Let $i$ be a key position where $x$ occurs in $A$ and $j$ be a position where $x$ occurs in $B$. By hypothesis, $e[i] = t[j]$ but by construction $f'[j]$ is a fresh new element. Hence $t = f$ as desired. □

Thus, to prove Theorem 33 it is sufficient to show that the three properties of Proposition 35 are satisfied by a query admitting a triangle-fork $q$-connected database.

We first prove that if a query $q$ admits a triangle-TRIPATH then the first two conditions of Proposition 35 should hold.

**Proposition 36.** *Let $q = AB$ be a 2way-determined query. If $q$ admits a triangle-*TRIPATH *then both of the following conditions hold:*

1. *There exists $x \in key(A)$ such that for all key position $j$ if $A[j] = x$ then for all $(j,k) \in q\text{-}fix$ we have $A[k] \notin key(B)$.*
2. *There exists $z \in key(B)$ such that for all key position $j$ if $B[j] = z$ then for all $(j,k) \in q\text{-}fix$ we have $B[k] \notin key(A)$.*

Towards proving this proposition, we need some definitions and observations. Recall that an alternating path is of the form $a_0b_0a_1b_1 \dots a_nb_na_{n+1}$ where $a_i \sim b_i$ and $q\{b_ia_{i+1}\}$ and if $i \neq j$ then $a_i \not\sim a_j$. We say that $\pi = a_0b_0a_1b_1 \dots a_nb_na_{n+1}$ is weak alternating path if the last condition is relaxed. So in a weak alternating path it is possible that for some $i \neq j$ we have $a_i \sim a_j$ or it is also possible that $a_i = a_j$, $b_i = b_j$ or $a_i = b_j$ etc. We are only concerned with weak alternating paths in this proof. If $\Pi$ is a weak alternating path then let $\bar{\Pi}$ denote the reverse given by $b'a_{n+1}b_na_nb_{n-1}a_{n-1} \dots b_1a_1b_0$ where $b'$ is some arbitrary but fixed fact such that $b' \sim a_{n+1}$.

A weak alternating path $\Pi = a_0b_0a_1b_1 \dots a_nb_na_{n+1}$ is forward (backward) if for every $k \leq n$ we have $q(b_ka_{k+1})$ (for every $k \leq n$ we have $q(a_{k+1}b_k)$ respectively). We say that an weak alternating path $\Pi$ is unidirectional if $\Pi$ is either forward or backward. Further, for unidirectional alternating paths $\Pi = a_0b_0a_1b_1 \dots a_nb_na_{n+1}$, we define its weight as $\mathsf{wt}(\Pi) = n$. Note that if $\Pi$ is forward with $\mathsf{wt}(\Pi) = n$ then $\bar{\Pi}$ is a backward with $\mathsf{wt}(\bar{\Pi}) = n$ and vice-versa.

Given a weak alternating path $\Pi = a_0b_0a_1b_1 \dots a_nb_na_{n+1}$ we say that $0 < l < n+1$ is a flip index if $q(b_{l-1}a_l) \wedge q(a_{l+1}b_l)$ or $q(a_lb_{l-1} \wedge q(b_la_{l+1})$, i.e. $l$ is a position that falisfies unidirectionality. Let $l_1, l_2 \dots l_k$ be the flip indices of $\Pi$. Then we consider the decomposition of $\Pi$ into $(\Pi_0, \Pi_1, \Pi_2, \dots, \Pi_k)$ where $\Pi_0 = a_0b_0a_1b_1 \dots a_{l_1-1}b_{l_1-1}a_{l_1}$, for every $i < k$: $\Pi_i = a_{l_i}b_{l_i} \dots a_{l_{i+1}-1}b_{l_{i+1}-1}a_{l_{i+1}}$ and $\Pi_k = a_{l_k}b_{l_k} \dots a_nb_na_{n+1}$. Note that each $\Pi_i$ is unidirectional. Moreover every $\Pi_i$ is forward iff $\Pi_{i+1}$ is backward. We define weight of $\Pi$ as the tuple $\mathsf{wt}(\Pi) = (\mathsf{wt}(\Pi_0), \mathsf{wt}(\Pi_1), \dots, \mathsf{wt}(\Pi_k))$.

We call $A_0B_0A_1B_1 \dots A_nB_nA_{n+1}$ to be a **generalized forward $n$-path** if for every forward alternating path $a_0b_0 \dots a_nb_na_{n+1}$ there is a homomorphism from $A_0B_0A_1B_1 \dots A_nB_nA_{n+1}$ to $a_0b_0 \dots a_nb_na_{n+1}$. We fix one generalized forward $n$-path for each $n$, which we call $E_n$. For $n = 1$, $E_1 = A'AB$, where $A' \sim A$ and every non-key variable in $A'$ is fresh. For $n = 2$, $E_2 = P'P_0P_1P'_1P_2$ (from the generalized 2-path $P_0P_1P'_1P_2$ defined in Appendix E, where $P' \sim P$ and every non-key variable in $P'$ is fresh).

The proof of Proposition 36 goes as follows. From the definition, any branch of a triangle-TRIPATH of $q$ is an alternating path starting from a fact of the triangle and ending at a fact excluding an element from the triangle key. We first show that this path can be assumed unidirectional. We then show that any such unidirectional path can not exclude an element from the key unless (1) and (2) are true.

The first step, transforming an alternating path into an unidirectional one, is based on the following two lemmas that shorten the path if a change of direction satisfies some properties.

**Lemma 37.** *Let $\Pi = a_0b_0a_1b_1 \dots a_nb_na_{n+1}$ be a weak alternating path and let $\alpha$ be such that for all $i \leq n$ we have $\alpha \in key(a_i)$ and $\alpha \notin key(a_{n+1})$.*

*If $\Pi$ is decomposed into $(\Pi_0, \Pi_1, \dots \Pi_{k-1}, \Pi_k)$, where $\mathsf{wt}(\Pi_{k-1}) \geq \mathsf{wt}(\Pi_k)$, then there exists $\Pi' = c_0d_0c_1d_1 \dots c_md_mc_{m+1}$, where $c_0 = a_0$ and $\alpha \notin key(c_{m+1})$ such that $\Pi'$ has strictly less flip indices than $\Pi$.*

*Proof.* Since $\mathsf{wt}(\Pi_{k-1}) \geq \mathsf{wt}(\Pi_k)$ let $\Pi'_{k-1} = a_sb_sa_{s+1}b_{s+1} \dots a_tb_ta_{t+1}$ be the suffix of $\Pi_{k-1}$ where $\mathsf{wt}(\Pi'_{k-1}) = \mathsf{wt}(\Pi_k) = t - s$. Assume that $\Pi_{k-1}$ (and hence $\Pi'_{k-1}$) is forward and $\Pi_k$ is backward (the other case is symmetric). Hence there is a homomorphism $h_1$ from $E_{t-s} = A'B_0A_1B_1 \dots A_{t-s}B_{t-s}A_{t-s+1}$ to $\Pi'_{k-1}$ and a homomorphism $h_2$ from $E_{t-s}$ to $\bar{\Pi}_k$. Notice that $h_1(A_{t-s+1}) \sim h_2(A_{t-s+1})$, thus $h_1$ and $h_2$ agree on $key(A_{t-s+1})$.

Now define a new homomorphism $h$ where for every $x \in vars(E_{t-s})$ if $x \in key(B_0)$ then $h(x) = h_1(x)$ and otherwise $h(x)$ is fresh.

Let $h(E_{t-s}) = c_s d_s c_{s+1} d_{s+1} \dots c_t d_t c_{t+1}$ which is a forward alternating path. Clearly $c_s \sim a_s$. We define the new weak alternating path $\Pi'$ as the one which is decomposed into $(\Pi_0, \Pi_1, \dots, \hat{\Pi}_{k-1})$, where $\hat{\Pi}_{k-1}$ is obtained from $\Pi_{k-1}$ by replacing $b_s a_{s+1} b_{s+1} \dots a_t b_t a_{t+1}$ with $d_s c_{s+1} d_{s+1} \dots c_t d_t c_{t+1}$. Clearly $\Pi'$ has strictly less flip indices than $\Pi$; moreover notice that $a_0$ is never replaced, thus by construction $\Pi'$ starts with the fact $a_0$.

It only remains to prove that $\alpha \notin key(c_{t+1})$. Towards this, let $X \subseteq key(B_0)$ be the set of variables such that for every $x \in X$ we have $h_1(x) = \alpha$. So if $X \cap key(A_{t-s+1}) = \emptyset$ then we are done. Suppose $x \in X \cap key(A_{t-s+1})$ then $h_2(x) = \alpha$. This implies $\alpha \in key(a_{n+1})$ which is a contradiction. □

**Lemma 38.** *Let $\Pi = a_0 b_0 a_1 b_1 \dots a_n b_n a_{n+1}$ be a weak alternating path such that $\Pi$ is decomposed into $(\Pi_0, \Pi_1, \dots \Pi_{k-1}, \Pi_k)$.*

*If there exists $0 < i < k$ such that $\mathsf{wt}(\Pi_{i-1}) \geq \mathsf{wt}(\Pi_i)$ and $\mathsf{wt}(\Pi_{i+1}) \geq \mathsf{wt}(\Pi_i)$ then there exists $\Pi' = c_0 d_0 c_1 d_1 \dots c_m d_m c_{m+1}$ where $c_0 = a_0$ and $c_{m+1} \sim a_{n+1}$ where $\Pi'$ has strictly less flip indices than $\Pi$.*

*Proof.* Let $\mathsf{wt}(\Pi_i) = k$ and assume $\mathsf{wt}(\Pi_{i-1}) \geq k$ and $\mathsf{wt}(\Pi_{i+1}) \geq k$. Consider the suffix $\Pi'_{i-1}$ of $\Pi_{i-1}$ of weight $k$ and the prefix $\Pi'_{i+1}$ of $\Pi_{i+1}$ of weight $k$.

Assume that $\Pi_{i-1}$ and $\Pi_{i+1}$ are forward while $\Pi_i$ is backward (the other case is symmetric). So there is a homomorphism $h_1$ from $E_k = A' B_0 A_1 B_1 \dots A_k B_k A_{k+1}$ to $\Pi'_{i-1}$ and a homomorphism $h_2$ from $E_k$ to $\Pi'_{i+1}$. There is also a homomorphism $h_3$ from $E_k$ to $\bar{\Pi}_i$.

Let $h_1(B_0) = b$ and $h_1(A_{k+1}) = a$ be the starting and ending facts of $\Pi'_{i-1}$. Similarly let $h_2(B_0) = b'$ and $h_1(A_{k+1}) = a'$ be starting and ending facts of $\Pi'_{i+1}$. Also let $h_3(B_0) = u$ and $h_3(A_{k+1}) = v$ be the starting and ending facts of $\Pi_i$. This implies by construction that $a \sim v$ and $b' \sim u$.

Define a new homomorphism $h$ where for every $x \in vars(E_k)$ if $x \in key(B_0)$ then $h(x) = h_1(x)$ and otherwise $h(x) = h_2(x)$. Let $h(E_k) = c_0 d_0 c_1 d_1 \dots c_k d_k c_{k+1}$. Clearly $d_0 \sim b$.

Now we claim that $c_{k+1} \sim a'$. To see this, let $y \in key(A_{k+1})$ we prove that $h(y) = h_2(y)$. If $y \notin key(A_0)$ then the claim follows by definition. Otherwise let $i, j$ be the key positions such that $A_0[i] = A_{k+1}[j] = y$. Then we have $h(y) = h_1(y) = d_0[i] = b[i] = a[j] = v[j] = u[i] = b'[i] = a'[j] = h_2(y)$.

The new alternating path $\Pi'$ is obtained from $\Pi$ by replacing $\Pi'_{i-1} \Pi_i \Pi'_{i+1}$ by $c_0 d_0 c_1 d_1 \dots c_k d_k c_{k+1}$. Note that $\Pi'$ is an alternating path since $c_0$ and $c_{k+1}$ are in the intended blocks and $\Pi'$ has strictly less flip indices. □

We are now ready to conclude the first step, transforming an alternating path into a unidirectional one.

**Lemma 39.** *Let $q$ be a query that is 2way-determined. If $q$ admits a triangle* TRIPATH *then we can construct a unidirectional weak alternating path $a_0 b_0 \dots a_n b_n a_{n+1}$ where $a_0 \in \{d, e, f\}$ such that $def$ is a triangle and $key(a_0) \not\subseteq key(a_{n+1})$.*

*Proof.* Let $\Theta$ be a triangle TRIPATH centered at the triangle $def$. Let $\Pi = a_0 b_0 \dots a_n b_n a_{n+1}$ be a weak alternating path with minimal number of flips where $a_0 \in \{d, e, f\}$ and $key(a_0) \notin key(a_{n+1})$. Since $\Theta$ is a triangle-TRIPATH, there such a $\Pi$ and hence a minimal one.

Now suppose $\Pi$ is unidirectional then we are done. Otherwise let $\Pi$ be decomposed into $(\Pi_0, \Pi_1, \dots \Pi_k)$ and $\mathsf{wt}(\Pi) = (w_0, \dots w_k)$ where $k \geq 1$.

We claim that for every $0 < i$ we have $w_{i-1} < w_i$. Suppose not then $i$ be the largest index where $w_{i-1} \geq w_i$. If $i = k$ then we have $w_{k-1} \geq w_k$ then by Lemma 37 we can obtain another alternating path with strictly less flip indices which contradicts the minimality of $\Pi$. If $i < k$ then we have $\mathsf{wt}(\Pi_{i-1}) \geq \mathsf{wt}(\Pi_i)$ and $\mathsf{wt}(\Pi_i) < \mathsf{wt}(\Pi_{i+1})$. In this case, by Lemma 38 we can get another alternating path with strictly less flips than $\Pi$ which again contradicts the minimality of $\Pi$. So we have $w_0 < w_1 < \dots < w_k$.

Assume that $\Pi_0$ is a forward alternating path (the case where $\Pi_0$ is backward alternating path is symmetric) and without loss of generality let $a_0 = e$. Since $w_0 < w_1$ let $w > 0$ such that $w + w_0 = w_1$.

Now consider the weak alternating path $\Pi' = e_0e_0f_0f_0d_0d_0e_1e_1f_1f_1d_1d_1 \dots e_we_wf_wf_wd_wd_wa_0b_0 \dots a_nb_na_{n+1}$. Then the new prefix has the same direction as $\Pi_0$. Hence $\Pi'$ has the same number of flip indices as $\Pi$ and $\Pi'$ is decomposed into $(\Pi'_0, \Pi_1, \dots \Pi_k)$ with $\mathsf{wt}(\Pi') = (w_1, w_1, w_2 \dots w_k)$ where $k \geq 1$.

But then, if $k = 1$ then $\mathsf{wt}(\Pi') = (w_1, w_1)$ so by Lemma 37 we can obtain another alternating path with strictly less flip indices which contradicts the minimality of $\Pi$. If $k > 1$ then $\mathsf{wt}(\Pi') = (w_1, w_1, w_2 \dots w_k)$ where $w_1 < w_2$. So by Lemma 38 we can get another alternating path with strictly less flips than $\Pi$ which again contradicts the minimality of $\Pi$. □

We now move on to the second step showing that assuming that either (1) or (2) is false then a an element on the key can not be excluded. The following lemma is a key property of q-fix over unidirectional path.

**Lemma 40.** *Let $q$ be a query that is 2way-determined and def be a triangle and $a_0 \in \{d, e, f\}$. Let $a_0b_0 \dots a_nb_na_{n+1}$ be a unidirectional weak alternating path such that for some $\alpha \in key(a_0)$ we have $\alpha \notin key(a_{n+1})$. Then: for all $(i, j) \in$ q-fix and for all $l \leq n$ if $a_l[i] = \alpha$ then $a_l[j] = \alpha$.*

*Proof.* Since the weak alternating path is unidirectional, assume it is forward, i.e. that for all $l < n + 1$ we have $q(b_la_{l+1})$. The other case is symmetric.

Now let $(i, j) \in \mathsf{q\text{-}fix}^k$ and assume $a_l[i] = \alpha$. We need to show that $a_l[j] = \alpha$. The case $l = 0$ is solved by Lemma 34. So assume $l > 0$. Since $\alpha \in key(a_l)$ we have $l < n + 1$. So we have $q(b_{l-1}a_l) \wedge q(b_la_{l+1})$.

We do an induction on $k$. In the base case $(i, j) \in \mathsf{q\text{-}fix}^0$. Then either $A[i] = A[j]$ or $B[i] = B[j]$. As $q(b_{l-1}a_l) \wedge q(b_la_{l+1})$, there is a homomorphism from both $key(A)$ and $key(B)$ to $key(a_l)$. Hence $a_l[j] = a_l[i] = \alpha$.

For the induction step let $(i, j) \in \mathsf{q\text{-}fix}^{k+1}$.

Now we consider various cases depending on $(i, j) \in \mathsf{q\text{-}fix}^{k+1}$.

- If there exists $(i_1, j_1) \in \mathsf{q\text{-}fix}^k$ such that $A[i] = B[i_1]$ and $A[j] = B[j_1]$ then $a_{l+1}[i_1] = b_l[i] = a_l[i] = \alpha$. By induction we get that $a_{l+1}[j_1] = \alpha$ and we conclude because $a_l[j] = b_l[j] = a_{l+1}[j_1]$.
- If there exists $(i_1, j_1) \in \mathsf{q\text{-}fix}^k$ such that $B[i] = A[i_1]$ and $B[j] = A[j_1]$ then $a_{l-1}[i_1] = b_{l-1}[i_1] = \alpha$ By induction we get that $a_{l-1}[j_1] = \alpha$ and we conclude because $a_l[j] = b_{l-1}[j_1] = a_{l-1}[j_1]$.
- If $(i, j) \in \mathsf{q\text{-}fix}^{k+1}$ because of transitive closure, then let $(i, j) \in \mathsf{q\text{-}fix}^{k+1}$ at step $s$ for some $s \geq 0$. We further induct on $s$. If $s = 0$ then we are in one of the previous two cases.

  Otherwise there exists $i'$ such that $(i, i') \in \mathsf{q\text{-}fix}^{k+1}$ at step $s - 1$ and $(i', j) \in \mathsf{q\text{-}fix}^{k+1}$ at step 0. Since $a_l[i] = \alpha$, inductively $a_l[i'] = \alpha$. But now $(i', j) \in \mathsf{q\text{-}fix}^{k+1}$ at step 0 and $a_l[i'] = \alpha$. So by the previous two cases we have $a_l[j] = \alpha$.

□

A sequence of (not necessarily distinct) indices $i_0j_0i_1j_0, \dots i_nj_n$ is called a fixing sequence if for all $l$, $(i_l, j_l) \in$ q-fix and $A[j_l] = B[i_{l+1}]$. We say $I$ is cyclic if $(j_n, j_{n'}) \in$ q-fix for some $n' < n$. We will be only interested in cyclic fixing sequences. The next immediate result gives a normalized way to denote cyclic fixing sequences.

**Lemma 41.** *If $I = i_0j_0i_1j_1, \dots i_nj_n$ is a cyclic fixing sequence then for every $(i_n, j_{n'}) \in$ q-fix the sequence $I' = i_0j_0i_1j_1, \dots i_nj_{n'}$ (where $j_n$ is replaced by $j_{n'}$) is also a cyclic fixing sequence. Hence there exists a cyclic fixing sequence $I' = i_0j_0, \dots i_nj_n$ and $m \leq n$ such that we have $i_mj_mi_{m+1}j_{m+1} \dots i_nj_n$ where $A[j_n] = B[i_m]$.*

Assume that (1) is false (the case where (2) is false is symmetric). This is equivalent to the following statement: For every key position $i$ there exists $(i, j) \in$ q-fix such that $A[j] \in key(B)$.

Hence for every key position $i$ there exists a cyclic fixing sequence $I = i_0j_0 \dots i_nj_n$. For every key position $i$, let $I_i$ denote some cyclic fixing sequence that starts at $i$. From Lemma 41 we can

assume that there is a subsequence of $I_i$ called the **loop** given by $\hat{I}_i = i_m j_m i_{m+1} j_{m+1} \dots i_n j_n$ where $(i_k, j_k) \in$ q-fix, $A[j_k] = B[i_{k+1}]$ and $A[j_n] = B[i_m]$ where $m \leq n$.

**Lemma 42.** *Let $q$ be a 2way-determined query. If for every key position $i$ there exists $(i,j) \in$ q-fix such that $A[i] = x$ and $A[j] \in key(B)$ then for every database $D$ and triangles $def$ in $D$ and $a \in \{d,e,f\}$, for every key position $i$ there exists some $s$ such that in the cyclic fixing sequence $I_i = i_0 j_0 \dots i_n j_n$ with the loop $\hat{I}_i = i_m j_m i_{m+1} j_{m+1} \dots i_n j_n$ such that for some $m \leq s \leq n$ we have $a[i_s] = a[j_s] = a[i]$.*

*Proof.* Let $a[i] = \alpha$. Denote $a^0 = a$, $a^1 = b \in \{def\}$ such that $q(ab)$ holds and $a^2 = \{d,e,f\} \setminus \{a, a^1\}$. For all $t \leq 3$ let $a^t = a^{t'}$ where $t' = t \mod 3$.

Then we have $a^0[i] = a^0[i_0] = a^0[j_0] = \alpha$ which implies that $a^1[i_1] = a^1[j_1] = \alpha$ and so on. In general we have $a^t[i_t] = a^t[j_t] = \alpha$ for $t \leq n$ and also if $a^t[j_n] = \alpha$ then $a^{t+1}[i_m] = \alpha$. Hence there will always be an index $m \leq s \leq n$ such that $a^0[i_s] = a^0[j_s] = \alpha$. □

*Proof of proposition 36.* Assume that (1) is false and there exists a triangle-TRIPATH $\Theta$ with $def$ as the center triangle. Then by Lemma 39 we have a unidirectional weak alternating path $a_0 b_0 \dots a_n b_n a_{n+1}$ where $a_0 \in \{d,e,f\}$ and $key(a_0) \not\subseteq key(a_{n+1})$. Let $i$ be the key position such that $a_0[i] = \alpha \notin key(a_{n+1})$.

We will arrive at a contradiction by proving that $\alpha \in key(a_{n+1})$. Let $\hat{I}_i = i_m j_m i_{m+1} j_{m+1} \dots i_n j_n$ be the loop part of $I_i$. From Lemma 42 there exists $m \leq s \leq n$ such that $a_0[i_s] = a_0[j_s] = \alpha$.

Now we prove that for every $l \leq n+1$ there exists some $m \leq t \leq n$ such that $a_l[i_t] = a_l[j_t] = \alpha$. The proof is by induction on $l$. When $l = 0$ we have just shown that $t = s$ does the job.

For the induction step assume that $a_l[i_t] = a_l[j_t] = \alpha$. If $l = n+1$ then we are done.

Otherwise let $q(b_l a_{l+1})$ (the case $q(a_{l+1} b_l)$ is symmetric). So we have $a_{l+1}[i_{t+1}] = \alpha$ where $t+1 = m$ if $t = n$. Now from Lemma 40, since $a_{l+1}[i_{t+1}] = \alpha$ and $\alpha \notin key(a_{n+1})$ and $(i_{t+1}, j_{t+1}) \in$ q-fix, it follows that $a_{l+1}[j_{t+1}] = \alpha$ and we are done. □

In view of Proposition 35 and Proposition 36, in order to prove Theorem 33 it remains to show that if there is a $q$-connected database $D$ containing both a triangle-TRIPATH, and a fork $abc$ then $q$ admits a one-sided fork-TRIPATH. Note that some of the lemmas in this section use the assumption that the query has uniform triangles, however this is without loss of generality, since if this is not the case, by Corollary 32 the conclusion of Theorem 33 immediately follows.

First we consider the special case when the fork and the center of the triangle-TRIPATH in the database share a block, we will then move to the case where they use mutually distinct blocks. A fork $abc$ and a triangle-TRIPATH with center $def$, are said to be *$q$-adjacent* if there exists $u \in \{a,b,c\}$ and $v \in \{d,e,f\}$ with $u \sim v$. In this case we use $abc$ together with one branch of the triangle-TRIPATH to construct the one-sided TRIPATH, as proved in the following lemma.

**Lemma 43.** *Let $q = AB$ be a query that is 2way-determined and has uniform triangles. If there exists a database containing a fork $q$-adjacent to a triangle-*TRIPATH *then $q$ admits a one-sided fork-*TRIPATH*.*

*Proof.* Let $D$ be a database containing a triangle-TRIPATH with triangle $def$ as the center which is $q$-adjacent to a fork.

Among all forks of $D$ having a block in common with $def$ we chose $abc$ so that the branching fact $b$ is in a block in common with $def$ and we let $u = b$, if such a fork exists. Otherwise we chose an arbitrary fork $abc$ sharing a block with $def$ and we chose $u$ arbitrarily as any fact of the fork which is in a block of $def$. In either case let $B$ be the block of the chosen fact $u$. Also let $v$ and $w$ the other two facts of $abc$ different from $u$.

Without loss of generality assume that $u \sim e$. Recall that $def$ is a center of a TRIPATH. There must exist a branch of this TRIPATH that does not intersect the block of $v$, nor the block of $w$ (otherwise if each of the three branches intersects the block of either $v$ or $w$, there must exist two branches of the TRIPATH intersecting the same block, which contradicts the definition of TRIPATH).

Let $z \in \{d, e, f\}$ be the starting point of the branch which does not intersect the blocks of $v$ and $w$; let $z\Pi$ be the strict alternating path formed by this branch. Let $u_z$ be the last fact of $z\Pi$.

Since $q$ has uniform triangles $key(d) = key(e) = key(f) = g(e)$, and therefore, by definition of TRIPATH, there exists $x \in key(e) \setminus key(u_z)$.

Let $\Pi'$ be a new alternating path connecting $u$ to $u_z$ defined as follows :

- $\Pi' := u\Pi$ if $z = e$
- $\Pi' := uez\Pi$ if $z \neq e$

Note that $\Pi'$ is an alternating path. In fact in the first case of its definition $u \sim z$, where $z$ is in the same block as the first fact of $\Pi$; in the second case $u \sim e$ and $D \models q\{ez\}$. Moreover all blocks of $\Pi'$ are pairwise distinct : in the first case blocks of $\Pi'$ are the same as blocks of $z\Pi$; in the second case $\Pi'$ contains additionally the block of $e$ which, by definition of TRIPATH, does not intersect with blocks of $z\Pi$.

Moreover $\Pi'$ does not intersect the blocks of $v$ and $w$, because $z\Pi$ does not, and the block of $u$ is distinct from the blocks of $v, w$ (as $u, v$ and $w$ form a fork).

Let $b_0 b_1 \dots b_n$ be the sequence of facts of $\Pi'$ with $b_0 = u$ and $b_n = u_z$. Note that $b_1 \sim b_0$, that $n \geq 2$ and $D \models q\{b_1 b_2\}$, since $\Pi'$ intersects at least two distinct blocks. We now prove that $\Pi'$ is strict; we only need to prove that $b_0 \neq b_1$ since we know that $z\Pi$ is strict. This is proved as a corollary of the following more general claim:

**Claim 44.** *For all $i = 1 \cdots n$, $D \not\models q\{bb_i\}$.*

*Proof of the claim.* Assume by contradiction that $D \models q\{bb_i\}$ with $i > 0$. Assume first that $i > 1$; then $b_i \not\sim u$. Moreover $b_i \not\sim v$ and $b_i \not\sim w$ because $\beta$ does not intersect the blocks of $v$ and $w$. Recall that $\{u, v, w\} = \{a, b, c\}$, then we have $D \models q\{bb_i\}$, $D \models q\{ba\}$, $D \models q\{bc\}$ with $a, c, b_i$ in three distinct blocks. Since $q$ is 2way-determined, this contradicts Lemma 7.

Assume now $i = 1$, so $D \models q\{bb_1\}$, then there are two cases to consider. The first case is that $u = b$ then $b \sim b_1$ and so in this case $b$ forms a solution with $b_1, a$ and $c$ which are in three distinct blocks (since they coincide with the blocks of the fork $abc$). Consider now the second case that $u \neq b$, then by the choice of $abc$ and $u$ we conclude that there exists no fork in $D$ whose branching fact is in a block of $def$. On the other hand we know $D \models q\{bb_1\}$ and $D \models q\{b_1 b_2\}$, with $b_2 \not\sim b$, because $b \in \{v, w\}$ and $\Pi'$ does not intersects the blocks of $v, w$. We then have that $bb_1 b_2$ is either a fork or a triangle. If it is triangle we have that $D \models q\{bb_2\}$ and we have already proved that this leads to a contradiction. If $bb_1 b_2$ is a fork, we have that $D$ contains a fork whose branching fact $b_1$ is in a block of $def$ (since $b_1 \sim u \sim e$); this contradicts the choice of $abc$ and of $u$. This concludes the proof of the claim. □

We can now show that $b_0 \neq b_1$, otherwise if $(u =) b_0 = b_1$ we have two cases : if $u \in a, c$ then $D \models q\{bb_1\}$; if $u = b$ then $D \models q\{bb_2\}$. Both conclusions contradict Claim 44. We cannot yet conclude that $abc$ forms a one-sided fork-TRIPATH with branch $\Pi'$, as the key inclusion condition may not be satisfied. We may in fact have $g(b) \subseteq key(u_z)$. To obtain a one sided fork-TRIPATH we then replace $abc$ by a new fork having the same properties as $abc$ w.r.t $\Pi'$, and additionally enjoying the key inclusion condition. To this end let $F_0 F_1 F_2$ be the most general fork of $q$. Assume without loss of generality that the domain of $F_0 F_1 F_2$ and the domain of $D$ are disjoint. Then there exists a homomorphism $h_F$ from $F_0 F_1 F_2$ to $abc$. Let $j \in \{0, 1, 2\}$ such that $h_F(F_j) = u$.

Since there are three forks in the triangle $def$, there are three homomorphisms from the most general fork to $def$; in particular let $h_T$ be the one mapping $F_j$ to $e$. Note that $h_F$ and $h_T$ must agree on $key(F_j)$, because $h_F(F_j) = u \sim e = h_T(F_j)$. So let $h$ be the mapping defined as $h_F$ (or equivalently $h_T$) on $key(F_j)$, and defined as the identity on the rest of the domain of $F_0 F_1 F_2$. Let $f_i = h(F_i)$ for $i = 0, 1, 2$, and denote as $D' = D \cup \{f_0, f_1, f_2\}$. Note that $f_j \sim u \sim e \sim b_1$, moreover, since $q$ is preserved under homomorphisms, we have both $D' \models q(f_0 f_1)$ and $D' \models q(f_1, f_2)$.

We now extend the mappings $h_F$ and $h_T$ to be the identity on all the domain of $D'$. This way we have $h_T(h(F_i)) = h_T(F_i)$, thus $h_T$ is also a homomorphism from $f_0 f_1 f_2$ to the triangle, with $h_T(f_j) = e$. Similarly $h_F(h(F_i)) = h_F(F_i)$ for $i = 0, 1, 2$; thus $h_F$ is also a homomorphism from

$f_0f_1f_2$ to $abc$ with $h_F(f_j) = u$. This implies that $f_0, f_1$ and $f_2$ must have pairwise distinct keys, because so do $a, b$ and $c$. Moreover $D \not\models q\{f_0f_2\}$ otherwise $abc$ would be a triangle. This allows us to conclude that $f_0f_1f_2$ is a fork. We now show that $\{f_0, f_1, f_2\} \cup \{b_1, \dots b_n\}$ is a one-sided fork-TRIPATH. First of all $f_jb_1 \dots b_n$ forms an alternating path from $f_j$ to $u_z$, because $f_j \sim b_1$, and $\Pi' = b_0, b_1 \dots b_n$ is an alternating path to $u_z$. To prove that it is strict we only need to prove that $f_j \neq b_1$ as the other inequalities follow from strictness of $\Pi'$. Assume by contradiction that $f_j = b_1$; recall that $h_F$ is the identity on the domain of $D'$ thus $h_F(f_j) = h_F(b_1) = b_1$ on the other hand we have already remarked that $h_F(f_j) = u(= b_0)$, then $b_1 = b_0$ this contradicts strictness of $\Pi'$.

We now prove that the alternating path $f_jb_1 \dots b_n$ does not intersect the blocks of the facts in $f_0f_1f_2$ other than $f_j$. Assume this is not the case, then there is some $b_i, i > 1$ such that $b_i \sim f_k$ for some $k \in \{0, 1, 2\}, k \neq j$. Thus, again because $h_F$ is the identity on the domain of $D'$, we must have $h_F(f_k) \sim f_k \sim b_i$. However recall that $h_F(f_k) \in \{w, v\}$; this implies that $\Pi'$ intersects the block of $w$ or $v$, which contradicts the properties of $\Pi'$ proved above.

It remains to prove that $g(f_1) \not\subseteq key(u_z)$. Assume by contradiction that $g(f_1) \subseteq key(u_z)$. Let $k \in \{0, 1, 2\}$ be such that $g(f_1) = key(f_k)$. Then $key(f_k)$ only contains elements of $D$. We now use the fact that $h_T$ is a homomorphism from $f_0f_1f_2$ to the triangle, then $h_T(f_k) \in \{d, e, f\}$; moreover $h_T$ is the identity on the domain of $D$, then $h_T(f_k) \sim f_k$. We thus have $g(f_1) = key(f_k) = key(e) = key(f) = key(d)$; this contradicts $key(e) \not\subseteq key(u_z)$.

This proves that $\{f_0, f_1, f_2\} \cup \{b_1, \dots b_n\}$ is a one-sided fork-TRIPATH and concludes the proof of the lemma. □

In the more general case that $q$ admits a triangle-fork $q$-connected database, we will use the connection between the fork and the triangle to obtain a one-sided TRIPATH. Let $D$ be a database that is $q$-connected containing a triangle-TRIPATH with center $def$ and let $D$ also contain a fork $abc$. Note that since $D$ is $q$-connected, there is an alternating path connecting $abc$ and $def$. Using this alternating path we show that we can construct a one-sided fork-TRIPATH. Towards this we will define the notion of the most general pattern that has a homomorphism to this alternating path and show the existence of one-sided fork-TRIPATH using this pattern.

First we need to intuitively *normalise* this altnernating path between the triangle and fork so as to be able to use it as a branch of a one-sided TRIPATH (in particular the blocks need to be pairwise distinct). This is done in the following technical lemma.

**Lemma 45.** *Let $q = AB$ be a query that is 2way-determined. If $q$ admits a triangle-fork $q$-connected database where no fork is $q$-adjacent to a triangle-*TRIPATH *then there exists a database containing a triangle $def$, a fork $abc$ and a strict alternating path $\Pi = a_0b_0 \dots a_nb_na_{n+1}$ satisfying all of the following conditions :*

- *there exists $v \in \{d, e, f\}$ such that $a_0 = v$;*
- *there exists $u \in \{a, b, c\}$ such that $a_{n+1} \sim u$ and $a_{n+1} \neq u$;*
- *$a, b, c, d, e, f, a_1, \dots a_n$ are in mutually distinct blocks.*

*Proof.* Let $D$ be a triangle-fork $q$-connected database. Then $D$ contains a triangle-TRIPATH with center $def$, and also contains a fork. Since $D$ is $q$-connected, each fork of $D$ is connected to $def$ via an alternating path. Let $abc$ be a fork of $D$ connected to $def$ via an alternating path of shortest length.

By our hypotheses $abc$ cannot be $q$-adjacent to the the triangle-TRIPATH; thus $a, b, c, d, e, f$ are in pairwise distinct blocks. Then the shortest alternating path connecting them is of length $> 0$ (i.e. it contains at least two blocks). Let $\Pi = a_0b_0 \dots a_nb_na_{n+1}$ be such alternating path, and let $u \in \{a, b, c\}$ and $v \in \{d, e, f\}$ such that $a_0 = v$ and $a_{n+1} \sim u$. We claim that

1. $a, b, c, d, e, f, a_1, \dots a_n$ are in pairwise distinct blocks;
2. $\Pi$ is strict and $u \neq a_{n+1}$.

(1) By definition of alternating path, $a_0, \dots a_{n+1}$ are in pairwise distinct blocks. Now assume by contradiction that $a_j \sim w$ for some $j$ where $1 \leq j \leq n$ and some $w \in \{a, b, c, d, e, f\}$. If $w \in \{d, e, f\}$ then $a_j b_j \dots a_n b_n a_{n+1}$ forms an alternating path of length shorter than $\Pi$ connecting a block of $def$ to a block of $abc$. Similarly if $w \in \{a, b, c\}$ then $a_0 b_0 \dots a_{j-1} b_{j-1} a_j$ forms an alternating path of length shorter than $\Pi$ connecting a block of $def$ to a block of $abc$. In both cases we reach a contradiction thus $a, b, c, d, e, f, a_1, \dots a_n$ are in pairwise distinct blocks.

(2) Let $b_{n+1}$ denote $u$, and $a_{n+2}$ denote an arbitrary element of $abc$ such that $D \models q\{b_{n+1}a_{n+2}\}$. Then it suffices to show that for all $i$ where $0 \leq i \leq n+1$ we have $a_i \neq b_i$. For $i = 0$ we must have $a_0 \neq b_0$ otherwise $a_0 (= v)$ forms a solution with facts from three different blocks : $a_1$ and two distinct facts of the triangle; these are in three different blocks by (1). This would contradict that $q$ is 2way-determined.

Now assume by contradiction that there exists $i, 1 \leq i \leq n+1$ such that $a_i = b_i$. Notice that $D \models q\{b_{i-1}a_i\}$; but the equality $a_i = b_i$ implies $D \models q\{a_i a_{i+1}\}$. Since $b_{i-1}, a_i$ and $a_{i+1}$ are in pairwise distinct blocks they form either a fork or a triangle.

If $b_{i-1}a_i a_{i+1}$ forms a fork, then $a_0 b_0 \dots a_{i-1}$ forms an alternating path of strictly shorter length than $\Pi$, connecting a fact of $def$ to a fork. This contradicts the fact that $\Pi$ is the shortest alternating path connecting $def$ to a fork of $D$.

If $b_{i-1}a_i a_{i+1}$ forms a triangle, we first show that that in this case we must have $i \leq n$. Let $w$ be the branching fact in $abc$, notice that either $w = b_{n+1}$ of $w = a_{n+2}$. By contradiction if $a_{n+1} = b_{n+1}$ and $b_n a_{n+1} a_{n+2}$ forms a triangle, then both $b_{n+1}$ and $a_{n+2}$ form a solution with $b_n$, thus $D \models \{w b_n\}$; this is a contradiction as $q$ is 2way-determined. Thus $1 \leq i \leq n$. By removing the block of $a_i b_i$ from $\Pi$ we have $\Pi' = a_0 b_0 \dots a_{i-1} b_{i-1} a_{i+1} b_{i+1} \dots a_{n+1}$ which is still an alternating path connecting $def$ to $abc$ because $D \models q\{b_{i-1}a_{i+1}\}$. This is a contradiction since $\Pi'$ is strictly shorter than $\Pi$. This proves the lemma. □

Note that $F_0F_1F_2$ is the most general fork of the query $q = AB$. On similar lines we define $T_0T_1T_2$ to be the most general triangle of $q$ which is constructed as follows: For all $x, y \in vars(F_0) \cup vars(F_1) \cup vars(F_2)$ define $x \equiv y$ if any of the following holds:

- There exists positions $i, j$ such that $A[i] = A[j]$ and $F_2[i] = x$ and $F_2[j] = y$
- There exists positions $i, j$ such that $B[i] = B[j]$ and $F_0[i] = x$ and $F_0[j] = y$
- There exists positions $i, j$ such that $A[i] = B[j]$ and $F_2[i] = x$ and $F_0[j] = y$

For each equivalence class of variables, pick a representative. Then $T_0T_1T_2 = h(F_0F_1F_2)$ where for every variable $x \in vars(F_0) \cup vars(F_1) \cup vars(F_2)$ we have $h(x) = \hat{x}$ where $\hat{x}$ denotes the representative of the equivalence class of $x$. We call $T_0T_1T_2$ the *triangle of* $q$.

The following is the missing step to the proof of Theorem 33.

**Proposition 46.** *Let $q = AB$ be a query that is 2way-determined and has uniform triangles. If $q$ admits a triangle-fork $q$-connected database, then $q$ admits a one-sided fork-*TRIPATH*.*

*Proof.* Assume that $q$ admits a triangle-fork $q$-connected database $D$. If $D$ contains a fork $q$-adjacent to a triangle-TRIPATH we conclude using Lemma 43. Otherwise, by Lemma 45, there exists a database containing a triangle $d_0d_1d_2$, a fork $f_0f_1, f_2$ and a strict alternating path connecting the two. Without loss of generality let $d_0$ and $f_0$ be the facts connected by the alternating path given by the sequence $d_0c_1e_1c_2e_2 \cdots c_ne_n$ with $e_n \sim f_0$; also let $e_0 := d_0$ and $c_{n+1} := f_0$.

By Lemma 45 we have that $n > 0$, that $d_0, d_1, d_2, f_0, f_1, f_2, e_1, \dots e_{n-1}$ are in pairwise distinct blocks, and that $e_i \neq c_{i+1}$ for all $i \geq 1$.

Let $O$ be the database consisting of exclusively the facts $\{d_0, d_1, d_2, f_0, f_1, f_2\} \cup \{c_i, e_i | i = 1..n\}$. Therefore $O \models q(d_j, d_{(j+1) \bmod 3})$ for $j = 0, 1, 2$, and $O \models q\{c_ie_i\}$ for all $i = 1..n$.

Note that because $\{f_0, f_1, f_2\}$ is a fork, there exists exactly one $j \in \{0, 1, 2\}$ such that $O \not\models q(f_j, f_{(j+1) \bmod 3})$. Note also that the only equivalences among facts of $O$ are : $e_i \sim c_{i+1}$ for all $i \geq 0$

We now construct an instance $O'$ which generalizes $O$. Let the most general fork of $q$ be $F_0F_1F_2$, where atoms are ordered such that $F_0F_1F_2$ maps homomorphically to $f_0f_1f_2$. Similarly

let $T_0T_1T_2$ be the most general triangle of $q$, ordered such that $T_0T_1T_2$ has a homomorphism to $d_0d_1d_2$. Without loss of generality assume that $(\cup_{j=0,1,2} vars(T_j)) \cap (\cup_{j=0,1,2} vars(F_j)) = \emptyset$.

Moreover for each $c_i, e_i$ in $O$, $i \geq 1$, we introduce a new fresh copy of $AB$ (using new fresh variables), we denote this copy by $C_iE_i$ in such a way that $C_iE_i$ has a homomorphism to $c_ie_i$.

Then we define the new instance $O'$ as the set of atoms $\{T_i | i = 0, 1, 2\} \cup \{C_i, E_i | 1 \leq i \leq n\} \cup \{F_i | i = 0, 1, 2\}$. For uniformity of notation, we also set $E_0 := T_0$ and $C_{n+1} := F_0$.

Clearly there exists a homomorphism $\mu$ from $O'$ to $O$ which maps $T_j$ and $F_j$ to $d_j, f_j$ respectively, for all $j = 0, 1, 2$ and maps $C_i, E_i$ to $c_i, e_i$, for all $1 \leq i \leq n$. Notice that $\mu$ induces a bijection between facts of $O'$ and $O$. Notice that in $O'$ we do not have $E_i \sim C_{i+1}$.

Recall that $R$ is the relation symbol of both facts $A$ and $B$ of $q$ and that the first $k$ positions of $R$ forms a key. We introduce a new schema consisting of the relation symbol $S$ with $arity(S) = arity(R) + 1$. Intuitively the extra attribute in $S$ (its first attribute) encodes a block identifier, and the subsequent attributes copy the attributes of $R$. The relation schema $S$ has an associated functional dependency $\Sigma = \{A_0 \to A_1 \dots A_k\}$ where $A_0$ denotes the first attribute of $S$, and $A_1 \dots A_k$ denote the next $k$ attributes (which copy the key of $R$)[5].

We encode $O$ of schema $R$ under the new signature $S$, by associating a unique identifier to each block of $O$. For each fact $R(\bar{a})$ of $O$ belonging to the block $B$ of $O$, we construct the fact $S(l, \bar{a})$ where $l$ is the identifier of $B$. The instance thus obtained form $O$ will be denoted by $O^S$.

We also encode $O'$ under schema $S$ as follows : for each block $B$ of $O$, of identifier $l$, we introduce a new fresh element $w_l$. For each fact $L = R(\bar{x})$ in $O'$, let $l$ be the identifier of the block of $\mu(L)$, and let $L^S$ be the fact $S(w_l, \bar{x})$. We denote by $O'^S$ the instance $\{L^S | L \in O'\}$. Notice that the only pairs of distinct facts of $O'^S$ agreeing on attribute $A_0$ are the pairs $E_i^S, C_{i+1}^S$ for all $0 \leq i \leq n$.

We extend the mapping $\mu$ to the new variables by defining $\mu(w_l) = l$. This way $\mu$ is a homomorphism from $O'^S$ to $O^S$, and induces a bijection between facts of these two instances.

Note that $O'^S$ does not satisfy the functional dependencies $\Sigma$ defined on $S$, thus we can apply the *chase* procedure to enforce them. The reader is referred to [AHV95] for a definition of the chase; here we will only use the well known properties of it summarized in the following claim. This claim rephrases in our terms Lemma 8.4.17 of [AHV95], which relates an arbitrary instance to the result of its chase using functional dependencies. In particular the following claim results from the application of Lemma 8.4.17 of [AHV95] to our instance $O'^S$ and our functional dependency $\Sigma$.

**Claim 47** (from [AHV95], Chapter 8.)**.** *Let $P^S$ be the result of chasing the instance $O'^S$ with respect to the functional dependencies $\Sigma$. Then $P^S$ is another instance of schema $S$ having the following properties:*

1. *$P^S$ satisfies $\Sigma$.*
2. *There exists a homomorphism $\theta : vars(O'^S) \to vars(O'^S)$ such that $P^S = \theta(O'^S)$, and $\theta$ is the identity on variables $w_l$, for all block identifier $l$.*
3. *If $D$ is any instance of schema $S$ satisfying $\Sigma$, and $\nu$ is a homomorphism $\nu : O'^S \to D$ then $\nu = \nu \circ \theta$, and therefore $\nu$ is also a homomorphism $\nu : P^S \to D$ with $\nu(P^S) = \nu(O'^S)$.*

Notice that $O^S$ satisfies $\Sigma$ and has a homomorphism $\mu$ from $O'^S$, thus applying Claim 47, we obtain that $\mu = \mu \circ \theta$ and therefore $\mu$ is a homomorphism from $P^S$ to $O^S$ with $\mu(P^s) = \mu(O'^S) = O^S$.

This implies that, since $\mu$ induces a bijection between facts of $O'^S$ and $O^S$, so does $\mu$ between $P^S$ and $O^S$, as well as $\theta$ between $O'^S$ and $P^S$. In other words $\theta$ cannot collapse any two distinct atoms of $O'^S$, as well as $\mu$ with $P^S$.

[5]A functional dependency over a relation schema $S$ is an expression of the form $X \to Y$, where $X, Y$ are sets of attributes of $S$. An instance $D$ of schema $S$ is said to satisfy the functional dependency $X \to Y$, denoted $D \models X \to Y$, if whenever $D$ contains facts $S(t_1)$ and $S(t_2)$, if tuples $t_1$, $t_2$ are identical on attributes $X$, then they are also identical on attributes $Y$.

**Claim 48.** *Given two facts $G, G'$ of $P^S$, the following are equivalent:*

*(a) $G, G'$ agree on attributes $A_1, \dots A_k$*

*(b) $\mu(G), \mu(G')$ agree on attributes $A_1, \dots A_k$*

*(c) $G, G'$ agree on attribute $A_0$*

*(d) either $G = G'$ or $\{G, G'\} = \{\theta(E_i^S), \theta(C_{i+1}^S)\}$ for some $0 \leq i \leq n$*

*Proof of the claim.* We first prove (a)⇒ (b)⇒(c)⇒(a).

Assume $G, G'$ agree on attributes $A_1, \dots A_k$, then so do facts $\mu(G), \mu(G')$. These are two facts of $O^S$, thus by construction of $O^S$, we have that $\mu(G)$ and $\mu(G')$ agree on attribute $A_0$. Now recall that $\mu$ is injective on block variables $w_l$, thus $G$ and $G'$ must agree on attribute $A_0$. Using the fact that $P^S$ satisfies $\Sigma$, this implies that $G, G'$ also agree on attributes $A_1, \dots A_k$.

It is easy to see that (c)⇔(d). In fact assume $G \neq G'$, then $G = \theta(H)$ and $G' = \theta(H')$, for some distinct facts $H, H'$ of $O'^S$. Since $\theta$ is the identity on block variables, $G, G'$ agree on attribute $A_0$ iff $H, H'$ do. By construction of $O'^S$ this happens iff $\{H, H'\} = \{E_i^S, C_{i+1}^S\}$ for some $0 \leq i \leq n$. □

Let $P$ denote the result of projecting out attribute $A_0$ from $P^S$, in the same way as $O'$ is obtained from $O'^S$ and $O$ from $O^S$. Then $P$ can also be viewed as an instance of schema $R$.

We now show that $P$ has the same 'shape' as $O$, *i.e.* it consists of a triangle connected to a fork via an alternating path. However $P$ is more 'general' than $O$ in the sense that it maps homomorphically to it; we will then show that $P$ has an endomorphism to its triangle.

Because homomorphisms are preserved when projecting out one attribute, we have that $\mu$ is a homomorphism from $O'$ to $O = \mu(O')$, as well as from $P$ to $O = \mu(P)$, and $\theta$ is a homomorphism from $O'$ to $P = \theta(O')$; moreover each of these homomorphisms induce a bijection between facts of the corresponding instances (this is because by construction $\mu$ induces a bijection between $O'$ and $O$ and $\mu = \mu \circ \theta$).

By construction $P = \{\theta(T_i) | i = 0, 1, 2\} \cup \{\theta(C_i), \theta(E_i) | 1 \leq i \leq n\} \cup \{\theta(F_i) | i = 0, 1, 2\}$. These are all pairwise distinct facts by the observation above that $\theta$ induces a bijection between facts of $T$ and $P$. Moreover using Claim 48 we have the following:

**Claim 49.** *For two distinct facts $N, N'$ of $P$, $N \sim N'$ iff $\{N, N'\} = \{\theta(E_i), \theta(C_{i+1})\}$ for some $0 \leq i \leq n$.*

*Proof.* Assume $\{N, N'\} = \{\theta(E_i), \theta(C_{i+1})\}$. These two facts can be obtained by projecting $A_0$ out from respectively $\theta(E_i^S)$ and $\theta(C_{i+1}^S)$, which by Claim 48 agree on attributes $A_1, \dots, A_k$. Thus $N$ and $N'$ agree on the key attributes. If conversely $N \sim N'$ then they are obtained by projecting out attribute $A_0$ from some distinct facts $G, G'$ of $P^S$, and $G, G'$ agree on attributes $A_1, \dots, A_k$. So by Claim 48 $\{G, G'\} = \{\theta(E_i^S), \theta(C_{i+1}^S)\}$ for some $i$. Then $\{N, N'\} = \{\theta(E_i), \theta(C_{i+1})\}$. □

Thus, the blocks of $P$ are $\{\theta(T_1)\}, \{\theta(T_2)\}, \{\theta(F_1)\}, \{\theta(F_2)\}$ and $\{\theta(E_i), \theta(C_{i+1})\}$ for all $0 \leq i \leq n$ (recall that $E_0 = D_0$ and $C_{n+1} = F_0$).

Now recall that $O' \models q(T_j, T_{(j+1) \bmod 3})$ for all $j = 0, 1, 2$, therefore $P \models q(\theta(T_j), \theta(T_{(j+1) \bmod 3})$, for all $j = 0, 1, 2$. Thus $\theta(T_j), j = 0, 1, 2$ forms a triangle in $P$.

Similarly, because there are two values of $j$ in $0, 1, 2$ such that $O' \models q(F_j, F_{(j+1) \bmod 3})$, the same values of $j$ are such that $P \models q(\theta(F_j), \theta(F_{(j+1) \bmod 3})$. For the remaining value of $j$, $P \models q(\theta(F_j), \theta(F_{(j+1) \bmod 3})$ is not possible otherwise $O = \mu(P) \models q(f_j, f_{(j+1) \bmod 3})$ for the corresponding $j$ and this is a contradiction, since $f_0 f_1 f_2$ is not a triangle in $O$. Hence $\theta(F_0), \theta(F_1), \theta(F_2)$ forms a fork in $P$.

Finally, because $O' \models q\{C_i E_i\}$ for all $1 \leq i \leq n$, we have that $P \models q\{\theta(C_i)\theta(E_i)\}$ for all $1 \leq i \leq n$. Together with Claim 49, this shows that $(\theta(E_i)\theta(C_{i+1}))_{0 \leq i \leq n}$ forms a strict alternating path connecting $\theta(T_0) (= \theta(E_0))$ to $\theta(F_0) (= \theta(C_{n+1}))$.

Let $X$ be the triangle contained in $P$, i.e. $X = \{\theta(T_j), j = 0, 1, 2\}$. We now show that $P$ has a homomorphism $h$ to $X$ and $h$ is the identity on the domain of $X$. Towards this we again use the encoding over signature $S$.

Let $X^S$ be the subinstance of $P^S$ consisting of the facts $\{\theta(T_j^S) \mid j = 0, 1, 2\}$.

**Claim 50.** *There exists a homomorphism from $O'^S$ to $X^S$ which agrees with $\theta$ on the domain of $\{T_j^S \mid j = 0..2\}$.*

*Proof.* Let $Y$ be the subinstance $\{T_j \mid j = 0, 1, 2\}$ of $O'$, and $Y^S$ be subinstance $\{T_j^S \mid j = 0, 1, 2\}$ of $O'^S$. We show that there exists a homomorphism $h$ from $O'^S$ to $Y^S$ which is the identity on the domain of $Y^S$. Since $\theta$ is a homomorphism from $Y^S$ to $X^S$, the composition $\theta \circ h$ satisfies the desired properties.

Assume wlog that the block variable of $T_j^S$ is $w_j$ for $j = 0, 1, 2$.

Let $O_i'^S$ be the subinstance $Y^S \cup \{C_l^S, E_l^S | 1 \leq l \leq i\}$ of $O'^S$. We first show that for all $0 \leq i \leq n$, $O_i'^S$ has a homomorphism to $Y^S$ which is the identity on the domain of $Y^S$.

This is clearly true for $i = 0$. Assume now that $0 < i \leq n$ and that $O_{i-1}'^S$ has a homomorphism $\phi$ to $Y^S$ with $\phi$ being the identity on the domain of $Y^S$. We know that $\phi(E_{i-1}^S) \in Y^S$, so let $\phi(E_{i-1}^S) = T_{k_0}^S$ for some $k_0 \in \{0, 1, 2\}$.

Let $w_{i_0}, w_{i_1}$ be the block variables of respectively $E_{i-1}^S$ (that is the same as for $C_i^S$) and $E_i^S$. Then $\phi(w_{i_0}) = w_{k_0}$.

Recall that $\{C_iE_i\}$ is isomorphic to $\{AB\}$. Then $\{C_iE_i\}$ has a homomorphism to the triangle $Y$. Indeed, because $Y$ is a triangle, for all $j \in \{0, 1, 2\}$ there exists a homomorphism from $\{C_iE_i\}$ to $Y$ mapping $C_i$ to $T_j$; we denote by $\psi$ the homomorphism from $\{C_iE_i\}$ to $Y$ mapping $C_i$ to $T_{k_0}$; we let $T_{k_1}$ be $\psi(E_i)$.

We extend $\psi$ to block variables by setting $\psi(w_{i_0}) = w_{k_0}$ and $\psi(w_{i_1}) = w_{k_1}$. Then $\psi$ is a homomorphism from $C_i^S E_i^S$ to $T_{k_0}^S T_{k_1}^S$. We can extend $\phi$ with $\psi$ since they agree on $w_{i_0}$ which is the only common variable in their domains. This extension of $\phi$ is then a homomorphism from $O_i'^S$ to $Y^S$, and is still the identity on the domain of $Y^S$.

This completes the induction and shows that $O_n'^S$ has a homomorphism $\phi$ to $Y^S$ which is the identity on the domain of $Y^S$.

A similar argument, detailed next, allows to extend $\phi$ to the entire $O'^S$. We know $\phi(E_n^S) \in Y^S$, so let $\phi(E_n^S) = T_{j_0}^S$ for some $j_0 \in \{0, 1, 2\}$. Let $w_{p_0}, w_{p_1}, w_{p_2}$ be the block variables of respectively $E_n^S$ (as well as $F_0^S$), $F_1^S$ and $F_2^S$. The $\phi(w_{p_0}) = w_{j_0}$.

Recall that $\{F_0F_1F_2\}$ is isomorphic to the fork of $q$, then $\{F_0F_1F_2\}$ has a homomorphism to the triangle $Y$, and in particular also a homomorphism $\rho$ mapping $F_0$ to $D_{j_0}$. We also let $D_{j_1} = \rho(F_1)$ and $D_{j_2} = \rho(F_2)$.

We extend $\rho$ to block variables by setting $\rho(w_{p_0}) = w_{j_0}$, $\rho(w_{p_1}) = w_{j_1}$ and $\rho(w_{p_2}) = w_{j_2}$. Then $\rho$ is a homomorphism from $\{F_0^S F_1^S F_2^S\}$ to $Y^S$. We can extend $\phi$ with $\rho$ since they agree on $w_{p_0}$ which is the only common variable in their domains. This extension of $\phi$ is then a homomorphism from $O'^S$ to $Y^S$, and is the identity on the domain of $Y^S$. □

Now notice that $X^S$ satisfies the functional dependencies $\Sigma$, because it is a subinstance of $P^S$, and $P^S \models \Sigma$. Then we can use Claim 47 to conclude that $h = h \circ \theta$ and therefore $h$ is a homomorphism from $P^S$ to $X^S$. Moreover $h(\theta(T_j^S)) = h(T_j^S) = \theta(T_j^S)$ for all $j \in \{0, 1, 2\}$. This proves that there exists a homomorphism $h$ from $P^S$ to its subinstance $X^S$, and $h$ is the identity on the domain of $X^S$.

By projecting out attribute $A_0$ we obtain that $h$ is a homomorphism from $P$ to its subinstance $X = \{\theta(T_j), j \in \{0, 1, 2\}\}$, and $h$ is the identity on the domain of $X$.

This can be used to conclude that for each fact $L \in P$, if $key(L)$ is contained in the domain of $X$, then $L \in X \cup \{\theta(C_1)\}$. In fact if $key(L)$ is contained in the domain of $X$ then $h(\overline{key}(L)) = \overline{key}(L)$. On the other hand $h(L) \in X$, and therefore there exists a fact in $X$ in the same block as $L$; by Claim 49, $L \in X$ or $L = \theta(C_1)$.

This is the key property that allows us to prove that $P \setminus X$ is a one-sided fork-TRIPATH.

First of all notice that $P \setminus X$ consists of a fork $\{\theta(F_j), j \in \{0,1,2\}\}$ and a strict alternating path $\theta(F_0), \theta(E_n)\theta(C_n) \dots \theta(E_1)\theta(C_1)$, which never intersects the blocks of $\theta(F_1), \theta(F_2)$.

Observe that if $L$ is in the fork of $P$, *i.e.* if $L \in \{\theta(F_j), j \in \{0,1,2\}\}$ then $L \notin X$. Now recall that we can assume $n > 0$, then $F_0 = C_{n+1} \neq C_1$ and therefore $\theta(F_j) \neq \theta(C_1)$ for all $j \in \{0,1,2\}$. Thus $key(\theta(F_j))$ is not contained in the domain of $X$ for all $j \in \{0,1,2\}$. By letting $k$ be such that $\theta(F_k)$ is the branching fact of the fork, we therefore have $g(\theta(F_k)) \not\subseteq key(\theta(C_1))$ (where recall that $\theta(C_1)$ is the ending point of the alternating path). This shows that $P \setminus X$ is a one-sided fork-TRIPATH. □

The proof of Theorem 33 now easily follows. Let $q = AB$ be a query that is 2way-determined and that admits a triangle-fork $q$-connected database. Since there exists a database containing a fork, $q$ is a real fork query. If $q$ does not have uniform triangles then by Corollary 32 $q$ admits a fork-TRIPATH. If $q$ has uniform triangles then, by Proposition 46, $q$ admits a one-sided fork-TRIPATH; therefore, by Proposition 35 and Proposition 36, $q$ admits a fork-TRIPATH.

We end the section by noting that there are queries like $q_7 = R(\underline{x_1x_2x_3\ y_1y_1y_2y_3\ z_1z_2z_3}\ z_4z_4z_4z_4) \wedge R(\underline{x_3x_1x_2\ y_3y_1y_1y_2\ z_2z_3z_4}\ z_1z_2z_3z_4)$ which admits a triangle-TRIPATH but every $q$ connected database that contains a triangle-TRIPATH does not contain any fork.

# H Proofs for Section 10 (Queries that admit only triangle-tripath)

**Theorem 14.** *Let $q$ be a 2way-determined query admitting a triangle-*TRIPATH*. Then for all $k$,* CERTAIN$(q) \neq$ Cert$_k(q)$.

The proof is essentially a reduction to the query $q_6 := E(\underline{x}yz) \wedge E(\underline{z}xy)$ for which it is shown that CERTAIN$(q_6)$ can not be solved using Cert$_k(q_6)$, for all $k$ [FPSS23].

It is actually known that CERTAIN$(q_6)$ cannot be solved using a small extension of the algorithm Cert$_k$ and we will make use of this fact: we show that if $q$ admit a triangle-TRIPATH and if CERTAIN$(q)$ = Cert$_k(q)$, then the extension of the algorithm Cert$_k$ solves CERTAIN$(q_6)$, which is a contradiction.

We now present the extension of Cert$_k$. Recall that in Cert$_k$, 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 $S' \in \Delta_k(q, D)$. In the extension, denoted $\Delta_k^+(q, D)$ we also add a $k$-set $S$ to $\Delta_k^+(q, D)$ if there exists a fact $a$ of $D$ such that for every fact $u \in D$, if $u = a$ or $D \models q\{au\}$ then there exists $S' \subseteq S \cup \{u\}$ such that $S' \in \Delta_k^+(q, D)$. It is easy to verify that the invariant, stating that any repair containing a set $S \in \Delta_k^+(q, D)$ makes the query true, is maintained. To see this consider a repair $r$ containing a set $S$ of fact constructed this way. If $r$ contains the fact $a$ or any fact $b$ making $q$ true with $a$, then by induction $r \models q$. Otherwise, let $r'$ be the repair constructed from $r$ by selecting $a$ for the block of $a$. By induction $r' \models q$. But by hypothesis this can not be because of $a$. Hence $r \models q$. As usual we accepts if the empty set is eventually derived. The resulting algorithm is denoted Cert$_k^+(q)$. As for Cert$_k(q)$ it runs in polynomial time and may only give false negative. We recall the result of [FPSS23]:

**Proposition 51.** *[FPSS23]* CERTAIN$(q_6)$ *cannot be computed by* Cert$_k^+(q_6)$*, for any choice of $k$.*

We now turn to the proof of Theorem 14.

Let $q$ be a query admitting a triangle-TRIPATH. We show that CERTAIN$(q)$ can not be solved using Cert$_k(q)$ no matter what $k$ is.

We show that if CERTAIN$(q)$ = Cert$_k(q)$ then Cert$_k^+(q_6)$ = CERTAIN$(q_6)$ contradicting Proposition 51.

Let $\Theta$ be a triangle-TRIPATH for $q$. Let $def$ be the center of $\Theta$. From Proposition 8 we can assume that $\Theta$ is a triangle-nice-TRIPATH. This means that there exists $x \in key(d), y \in key(e)$ and

$z \in key(f)$ such that $\{x,y,z\} \cap \big(key(B_r) \cup key(B_{l_1}) \cup key(B_{l_2})\big) = \emptyset$ and $q(\Theta) = \{(fd)\} \cup \{\{ab\} \mid a = a(B_i), b = b(B_j), s(B_i) = B_j\}$.

Also let $u \in key(B_r)$, $v \in key(B_{l_1})$ and $w \in key(B_{l_2})$ be the elements in the keys that do not occur in the key of any other blocks in $\Theta$.

For any elements $\alpha_x, \alpha_y, \alpha_z, \alpha_u, \alpha_v \alpha_w$, we denote by $\Theta[\alpha_x, \alpha_y, \alpha_z, \alpha_u, \alpha_v, \alpha_w]$ the database constructed from $\Theta$ by replacing $x, y, z$ by $\alpha_x, \alpha_y, \alpha_z$ and $u, v, w$ by $\alpha_u, \alpha_v, \alpha_w$ respectively (we will ensure that $\alpha_x = \alpha_y$ iff $x = y$ etc.)

We now describe the reduction. Let $D$ be a database for $q_6$. We construct from $D$ a database $D'$ for $q$ satisfying the following properties: $D \models \text{CERTAIN}(q_6)$ iff $D' \models \text{CERTAIN}(q)$ and moreover $D' \models \text{Cert}_k(q)$ implies $D \models \text{Cert}_k^+(q_6)$. This will conclude the proof of Theorem 14.

For the construction of $D'$ we split the facts of $D$ into cliques: Consider a fact $(\underline{u}\ vw)$ of $D$. If both $(\underline{w}\ uv)$ and $(\underline{v}\ wu)$ are facts of $D$ then those three facts form a clique of size three. If only one of $(\underline{w}\ uv)$ and $(\underline{v}\ wu)$ is present then it forms a clique of size two with $(\underline{u}\ vw)$. Otherwise $(\underline{u}\ vw)$ forms a clique of size one. It is straightforward to verify that only one of the three cases applies for every fact and $D \models q(ab)$ iff $a$ and $b$ belong to the same clique.

Let $C$ be a clique of $D$. Let $(\underline{u}\ vw)$ be a fact of $C$. Consider $\Theta[\langle u,v,w\rangle, \langle u,v,w\rangle, \langle u,v,w\rangle, u, v, w]$. Notice that $u$, $v$ and $w$ occur only in the key of one of the ending block. Remove from $\Theta$ the branch of $v$ if $(v,w,u) \notin C$. Similarly for $w$. Denote by $\theta_C$ the resulting database, denoted the gadget for $C$ in the sequel.

Let $D'$ be the union of $\Theta_C$ for all cliques $C$ of $D$. We claim that $D'$ has the desired properties.

We start with a few observations. Each fact $(\underline{u}\ vw)$ of $D$ is associated to a fact $f(\underline{u}\ vw)$ of $D'$: this is the fact of the endpoint of the gadget containing $(\underline{u}\ vw)$. Notice that the only new element in the key of $f(\underline{u}\ vw)$ is $u$, and $f(\underline{u}\ vw)$ contains at least one non key element $\langle u,v,w\rangle$. Hence each block of $D$ is associated to a block of $D'$ of the same size. The remaining blocks of $D'$ are of size 2 and are inner block of a gadget $\Theta_C$ for some clique $C$ of $D$. It is also easy to check that for each clique $C$ and each end point fact $a$ of $\Theta_C$, it is possible to select one fact per inner block of $\Theta_C$ such that the partial repair with all these facts together with $a$ does not make the query true. However this is not possible with 2 endpoints of $\Theta_C$. This implies that $D$ contains a repair falsifying $q_6$ iff $D$ contains a repair falsifying $q$.

It remains to show that if $D' \models \text{Cert}_k^+(q)$ then $D \models \text{Cert}_k(q_6)$. To this end it is enough to show that if a set $S'$ contains only facts of the form $f(a)$ for some fact $a$ of $D$ and $S' \in \Delta_k(q, D')$ then $f^{-1}(S') \in \Delta_k^+(q_6, D)$.

To prove this we need some extra notations. Consider a gadget $\Theta_C$ for some clique $C$ of $D$. It has three specific facts forming a triangle, and at most three 'branches' of blocks forming alternating paths starting from this triangle. Consider one of the branch of $\Theta_C$. We give a label 0 to the fact within the triangle and label 1 for the other fact within the same block. By induction, going block by block starting from the block connected to the triangle to the endpoint fact, we label 0 a fact connected to a fact of label 1 and label 1 the other fact of the block. Hence the ending point is labeled 0.

Let $a$ be a fact within an inner block of $\Theta_C$. We associate to $a$ via $g$ one or two of the endpoints of $\Theta_C$ as follows. If $a$ has label 0 then $g(a)$ contains the endpoint fact of the branch where $a$ is. If $a$ has label 1 then $g(a)$ contains the two other endpoint facts of $\Theta_C$. The key property of $g$ is the following lemma:

**Lemma 52.** *Let $S'$ be a set of facts of $D'$ and $a$ an inner fact of some gadget $\Theta_C$. If $S' \cup \{a\} \in \Delta_k(q, D')$ then for any fact $b \in g(a)$, $S' \cup \{b\} \in \Delta_k(q, D')$*

The proof follows by simple induction on the distance of the block from the block of $a$.

A typical application of Lemma 52 is with one of the initial sets $S$ of $\Delta_k(q, D')$. Such a set contains two inner facts of a gadget making the query true. They have different labels and hence it follows from the definition of $g$ and Lemma 52 that any pair of endpoints of a gadget $\Theta_C$ is eventually in $\Delta_k(q, D')$. This is exactly what we expect as they inverse image by $f$ belongs initially to $\Delta_k(q, D')$.

To conclude the proof we assume a normal form in the derivation of the sets in $\Delta_k(q, D')$. We assume that each time a new set $S$ is derived, if $S$ contains a inner fact $a$ then we immediately

apply the necessary derivation that replace $a$ by any element of $g(a)$ as guaranteed by Lemma 52. In a sense we view this process as an $\varepsilon$-step in the derivation:, as if the sets involved are inserted in $\Delta_k(q, D')$ at the same time.

The proof is then by induction on the number of steps needed to derive $S'$. Recall that we assume $S' = f(S)$ for some set $S$ of facts of $D$. We want to show that $S \in \Delta_k^+(q_6, D)$. Assume $S'$ is added to $\Delta_k(q, D')$ because of a block $B'$ of $D'$. If $B'$ is associated to a block $B$ of $D$, then this block can be used to show that $S$ belongs to $\Delta_k^+(q_o, D)$. Otherwise, $B'$ is an inner block of some gadget $\Theta_C$. By definition, for any fact $a$ of $B'$, $S' \cup \{a\}$ contains a set already in $\Delta_k(q, D')$. Notice that the two facts of $B'$ have different label hence, by Lemma 52, $S' \cup \{b\}$ contains a set already in $\Delta_k(q, D')$ for any endpoint fact $b$ of $\Theta_c$. By induction this implies that $S \cup f^{-1}(b)$ contains a set already in $\Delta_k^+(q_6, D)$. By the new rule of $\text{Cert}_k^+$, this implies that $S \in \Delta_k^+(q_6, D)$ as desired.

This concludes the proof of Theorem 14.

**Proposition 15.** *Let $q$ be a 2way-determined query and $D$ be a database. Then $D \models \neg$MATCHING$(q)$ implies $D \models$ CERTAIN$(q)$.*

*Proof.* Assume $D \not\models$ CERTAIN$(q)$, let $r$ be a repair such that $r \models \neg q$. For each block $B$ of $D$ let $r(B)$ be the fact of $B$ belonging to $r$. The MATCHING(q) algorithm on $D$ constructs $G(D, q)$ and $H(D, q) = (V_1 \cup V_2, E)$. Note that elements of $V_2$ form a partition of $D$, thus each fact $r(B)$ belongs to a unique element of $V_2$ which is *clique*$(r(B))$. Define $f : V_1 \to V_2$ such that each block $B \in V_1$ is mapped to *clique*$(r(B))$. We claim that $f$ is a witness function of a $V_1$-saturating matching for $H(D, q)$. In fact for every $B \in V_1$ we have $(B, f(B)) \in E$, as $B$ and $f(B)$ both contain $r(B)$ and $D \not\models q(r(B), r(B))$ (otherwise $r$ would contain a solution). Moreover $f$ is injective, otherwise if $f$ maps two distinct blocks to the same $C \in V_2$, then $C$ contains two distinct elements $a, b \in r$, where $a \not\sim b$ (one in each block). It follows that $C$ is a quasi-clique (because $C$ is not a singleton) then we have that $q\{ab\}$ holds and hence $r \models q$, a contradiction.

Since $f$ is injective, it is a $V_1$-saturating matching for $H(D, q)$, thus MATCHING$(q)$ outputs "yes". □

**Proposition 16.** *Let $q$ be a 2way-determined query and $D$ be a clique-database for $q$. Then $D \models \neg$MATCHING$(q)$ iff $D \models$ CERTAIN$(q)$. Therefore checking whether $D \models$ CERTAIN$(q)$ is in* PTIME.

*Proof.* By Proposition 15 it suffices to prove that $D \models$ MATCHING$(q)$ implies $D \not\models$ CERTAIN$(q)$. So assume that MATCHING(q) outputs "yes" on $D$. Then we know that there is a $V_1$-saturating matching of $H(D, q) = (V_1 \cup V_2, E)$, that is, an injective function $f : V_1 \to V_2$ such that $(B, f(B)) \in E$ for every $B \in V_1$ (i.e. for every block $B$ of $D$). By construction of $H(D, q)$ the edge $(B, f(B)) \in E$ implies that there exists a fact $b \in B$ such that $b \in f(B)$ and $D \not\models q(b, b)$. Let $r$ be a repair where for every block $B$ of $D$, $r$ contains such fact $b$ of $f(B)$; hence $r$ does not contain solutions of the form $(b, b)$. Since elements of $V_2$ form a partition of $D$, the chosen fact $b$ belongs to only one element of $V_2$, thus $f(B) =$ *clique*$(b)$. Then injectivity of $f$ implies that for every two distinct $b_1, b_2 \in r$, *clique*$(b_1) \neq$ *clique*$(b_2)$.

Now since $D$ is a clique-database, for all $a \in D$ we have that *clique*$(a)$ is the connected component of $a$ in $G(D, q)$ (by definition of *clique*, since this component is a quasi-clique). Therefore $D \not\models q\{b_1, b_2\}$, otherwise we would have *clique*$(b_1) =$ *clique*$(b_2)$.

This proves that $r$ contains no solution consisting of two distinct facts. Finally by construction of $H(D, q)$, for all $b \in r$ we have $r \not\models q(b, b)$, this proves that $r \not\models q$. □

**Proposition 19.** *Let $q$ be a 2way-determined query that does not admit a fork-*TRIPATH *and let $D$ be a database. There exists a partition $C_1, C_2, \ldots C_n$ of $D$ having all of the following properties :*

1. *for all $i$, $C_i$ does not contain a* TRIPATH *or $C_i$ is a clique-database for $q$.*
2. *$D \models$ CERTAIN$(q)$ iff there exists some $i$ such that $C_i \models$ CERTAIN$(q)$.*
3. *For all $k$, if $C_i \models \text{Cert}_k(q)$ for some $i$, then $D \models \text{Cert}_k(q)$.*

*4. If $D \models$ MATCHING$(q)$ then for all $i$ $C_i \models$ MATCHING$(q)$.*

Towards proving this, we first set up some definitions. Recall the definition of $q$-connected blocks and $q$-connected database introduced in Section G.2. Note that every database can be partitioned into disjoint sets of blocks such that each partition is a $q$-connected database (moreover, we can obtain such a partition of the database in polynomial time). If the database $D$ is partitioned into $C_1 \cup C_2 \dots C_n$ where each $C_i$ is a $q$-connected database, we call this the $q$-connected partition of $D$.

We will show that this partition satisfies the properties of Proposition 19. The most difficult property to prove is (1), on which we concentrate first.

It turns out that for the queries that we are interested in, each $q$-connected database is of two possible forms, each allowing CERTAIN$(q)$ to be computed efficiently. This is formalized in the following proposition whose proof relies on Theorem 33, which adresses its main technical difficulty.

**Proposition 53.** *Let $q$ be a 2way-determined query that does not admit a fork-*TRIPATH*. Let $D$ be a $q$-connected database. Then $D$ contains no* TRIPATH *or $D$ is a clique-database for $q$.*

*Proof.* Towards a contradiction, assume that $D$ is a $q$-connected database that contains a TRIPATH $\Theta$ and $D$ is not a clique-database of $q$. Since we have assumed that $q$ does not admit a fork-TRIPATH, it follows that $\Theta$ is a triangle-TRIPATH. But now since $D$ is not a clique-database, $D$ also contains a fork (otherwise $D$ is a clique-database by definition). Thus, $D$ contains a triangle-TRIPATH and also contains a fork. But then, by Theorem 33 this implies $q$ admits a fork-TRIPATH, a contradiction. □

We are now ready to prove Proposition 19.

*Proof.* Let $C_1 \cup C_2 \dots C_n$ be the the $q$-connected partition of $D$. It satisfies property (1) by Proposition 53. We now prove that this partition satisfies all the remaining required properties.

(2) Suppose $D \not\models$ CERTAIN$(q)$ then clearly every $C_i$ has a repair that makes the query false and hence $C_i \not\models$ CERTAIN$(q)$ for every $i$.

Conversely, suppose there is some $i$ such that $C_i \models$ CERTAIN$(q)$ then pick any repair $r$ of $D$, then $r$ induces a partial repair $r'$ over $C_i$ and by assumption $r' \models q$. Hence $r \models q$.

(3) Note that for all $k$, $\text{Cert}_k(q)$ has a form of monotonicity, that is for all databases $D_1, D_2$ having no key in common, if $D_1 \models \text{Cert}_k(q)$ then $D_1 \cup D_2 \models \text{Cert}_k(q)$. This is because blocks of $D_1$ are still blocks in $D_1 \cup D_2$, and therefore any derivation of $\text{Cert}_k(q)$ in $D_1$ is also a derivation in $D_1 \cup D_2$. Note also that for all $i \neq j$ and all $a \in C_i$ and $b \in C_j$ we have $a \not\sim b$, thus $\text{Cert}_k(q)$ is monotone w.r.t adding components of the partition. In particular if $C_i \models \text{Cert}_k(q)$ for some $i$, then $D \models \text{Cert}_k(q)$.

(4) Recall the MATCHING$(q)$ algorithm on input $D$ runs (and outputs the result of) bipartite matching on the bipartite graph $H(D, q) = (V_1, V_2, E)$ defined in Section 10.1. We show that $H(D, q)$ is the disjoint union of $H(C_i, q)$, i=1..n. First notice that each block $B \in V_1$, as well as each component $C \in V_2$, is contained in exactly one $C_j$. Moreover if $\{B, C\} \in E$ and $B \subseteq C_i$ and $C \subseteq C_j$ then $i = j$; in fact, since there exists $b \in B \cap C$, then $b \in C_i \cap C_j$, which implies $i = j$. Then each $\{B, C\} \in E$ is also an edge in $H(C_i, q)$ for some $i$.

This shows that $H(D, q)$ is the disjoint union of $H(C_i, q), i = 1..n$, and therefore bipartite matching outputs "yes" on $H(D, q)$ iff for all $i$ it outputs "yes" on $H(C_i, q)$.

□