Theory of Computing Systems
https://doi.org/10.1007/s00224-020-09985-6

Consistent Query Answering for Primary Keys in Datalog

Paraschos Koutris[1] · **Jef Wijsen**[2]

Published online: 30 June 2020

Abstract

We study the complexity of consistent query answering on databases that may violate primary key constraints. A repair of such a database is any consistent database that can be obtained by deleting a minimal set of tuples. For every Boolean query q, CERTAINTY(q) is the problem that takes a database as input and asks whether q evaluates to true on every repair. In Koutris and Wijsen (ACM Trans. Database Syst. **42**(2), 9:1–9:45, 2017), the authors show that for every self-join-free Boolean conjunctive query q, the problem CERTAINTY(q) is either in **P** or **coNP**-complete, and it is decidable which of the two cases applies. In this article, we sharpen this result by showing that for every self-join-free Boolean conjunctive query q, the problem CERTAINTY(q) is either expressible in symmetric stratified Datalog (with some aggregation operator) or **coNP**-complete. Since symmetric stratified Datalog is in **L**, we thus obtain a complexity-theoretic dichotomy between **L** and **coNP**-complete. Another new finding of practical importance is that CERTAINTY(q) is on the logspace side of the dichotomy for queries q where all join conditions express foreign-to-primary key matches, which is undoubtedly the most common type of join condition.

Keywords Conjunctive queries · Consistent query answering · Datalog · Primary keys

This article belongs to the Topical Collection: *Special Issue on Database Theory (ICDT 2019)*
Guest Editor: Pablo Baceló

This article extends an earlier, shorter version entitled "Consistent Query Answering for Primary Keys in Logspace" which was presented at the 22nd International Conference on Database Theory (ICDT 2019) [23] .

✉ Jef Wijsen
jef.wijsen@umons.ac.be

Paraschos Koutris
paris@cs.wisc.edu

[1] University of Wisconsin-Madison, Madison, WI, USA

[2] University of Mons, Mons, Belgium

1 Motivation

Consistent query answering (CQA) with respect to primary key constraints is the following problem. Given a database **db** that may violate its primary key constraints, define a repair as any consistent database that can be obtained by deleting a minimal set of tuples from **db**. For every Boolean query q, the problem CERTAINTY(q) takes a database as input and asks whether q evaluates to true on every repair of **db**. In this article, we focus on CERTAINTY(q) for queries q in the class sjfBCQ, the class of self-join-free Boolean conjunctive queries. For all Boolean first-order queries q, CERTAINTY(q) is in **coNP** and therefore can be solved by expressive formalisms like answer set programming [28], binary integer programming [19], and SAT solvers [11]. These solutions are, theoretically, an overkill when CERTAINTY(q) also belongs to a lower complexity class. In particular, given a query q in sjfBCQ, it is decidable [21] whether CERTAINTY(q) is in the low complexity class **FO**. Moreover, if the problem CERTAINTY(q) is in **FO**, then it is possible to construct a first-order query for solving CERTAINTY(q), which is also called a *consistent first-order rewriting for* q. This construction is detailed in [21, Section 5] and has already been implemented [31]. It has been observed, however, that a theoretically lower complexity may not manifest itself in practice [11].

In [21], the authors also show that for every query q in sjfBCQ, the problem CERTAINTY(q) is either in **P** or **coNP**-complete, and it is decidable (in polynomial time in the size of q) which of the two cases applies. The authors show how to construct a polynomial-time algorithm for CERTAINTY(q) when it does not lie on the **coNP**-hard side of the dichotomy. Unfortunately, unlike for consistent first-order rewritings, this construction is complex and does not tell us what language would be appropriate for implementing CERTAINTY(q) when it is in **P** $\setminus$ **FO**. In this article, we improve this situation: we show that if CERTAINTY(q) is in **P**, then it can be implemented in symmetric stratified Datalog, which has deterministic logspace data complexity [12]. We thus sharpen the complexity dichotomy of [21] as follows: for every query q in sjfBCQ, CERTAINTY(q) is either in **L** or **coNP**-complete. It is significant that Datalog is used as a target language, because this allows using optimized Datalog engines for solving CERTAINTY(q) whenever the problem lies on the logspace side of the dichotomy. Rewriting into Datalog is generally considered a desirable outcome when consistent first-order rewritings do not exist (see, e.g., [5, page 193]). It is also worth noting that the SQL:1999 standard introduced linear recursion into SQL, which has been implemented in varying ways in existing DBMSs [32]. Since the Datalog programs in this article use only linear recursion, they may be partially or fully implementable in these DBMSs.

Throughout this article, we use the term *consistent database* to refer to a database that satisfies all primary-key constraints, while the term *database* refers to both consistent and inconsistent databases. This is unlike most database textbooks, which tend to say that databases must always be consistent. The following definition introduces the main focus of this article; the complexity dichotomy of Theorem 1 is the main novel contribution of this article.

Definition 1 Let q be a Boolean query. Let $\mathcal{L}$ be some logic. A *consistent $\mathcal{L}$ rewriting for q* is a Boolean query P in $\mathcal{L}$ such that for every database **db**, P is true in **db** if and only if q is true in every repair of **db**. If q has a consistent $\mathcal{L}$ rewriting, then we say that CERTAINTY(q) is expressible in $\mathcal{L}$.

Theorem 1 *For every self-join-free Boolean conjunctive query q, the problem* CERTAINTY(q) *is either* **coNP***-complete or expressible in SymStratDatalog*$^{\text{min}}$. *Therefore,* CERTAINTY(q) *is either* **coNP***-complete or in* ***L****.*

The language *SymStratDatalog*$^{\text{min}}$ will be defined in Section 3; informally, the superscript min means that the language allows selecting a minimum (with respect to some total order) from a finite set of values. Since CERTAINTY(q) is **L**-complete for some queries q in sjfBCQ, the logspace upper bound in Theorem 1 is tight. The proof of Theorem 1 relies on novel constructs and insights developed in this article. Compared to [21], significant new contributions are the notion of *garbage set* and the helping Lemmas 8 and 12.

Our second significant result in this article focuses on consistent query answering for foreign-to-primary key joins. In Section 9, we define a subclass of sjfBCQ that captures foreign-to-primary key joins, which is undoubtedly the most common type of join. We show that CERTAINTY(q) lies on the logspace side of the dichotomy for *all* queries q in this class. Therefore, for the most common type of joins and primary key constraints, CQA is highly tractable, a result that goes against a widely spread belief that CQA would be impractical because of its high computational complexity.

Organization Section 2 discusses related work. Section 3 defines our theoretical framework, including the notion of *attack graph*. To guide the reader through the technical development, Section 4 provides a high-level outline of where we are heading in this article, including examples of the different graphs used. Section 5 introduces the notion of garbage set for a subquery. Informally, garbage sets contain facts which can never make the subquery hold true, and which therefore can be removed from the database without changing the answer to CERTAINTY(q). Section 6 introduces the notion of M-graph, a graph at the schema-level, and its data-level instantiation, called $\hookrightarrow$-graph. Section 7 focuses on cycles in the M-graph of a query, and shows that garbage sets for such cycles can be computed and removed in symmetric stratified Datalog. Section 8 contains the proof of our main theorem. That section first introduces a special subclass of sjfBCQ, called *saturated queries*, and argues that in the complexity classification of the problems in $\{\text{CERTAINTY}(q) \mid q \in \text{sjfBCQ}\}$, it suffices to focus on saturated queries, exploiting some good properties that are proper to saturated queries. In particular, Lemma 12 relates cycles in attack graphs to cycles in M-graphs, for saturated queries only. This is the last ingredient needed for the proof of our main theorem, which is given at the end of Section 8. Section 9 shows that foreign-to-primary key joins fall on the logspace side of the dichotomy. Finally, Section 10 summarizes in a single theorem the complexity classification of all problems in $\{\text{CERTAINTY}(q) \mid q \in \text{sjfBCQ}\}$ obtained over the years.

Expressibility in symmetric stratified Datalog (with or without aggregation) is shown by means of constructive proofs, which act as guides for a practical

implementation of our theoretical results. Such constructive proofs are given in the main body of this article. To lighten the reading, full proofs which are less essential to the practice of consistent rewriting have been moved to an appendix; some of these proofs are nevertheless sketched in the main body of this article. Moreover, to help readability, Appendix A contains a list of notations for easy reference.

2 Related Work

Consistent query answering (CQA) started with the seminal work by Arenas, Bertossi, and Chomicki [2], and is the topic of a monograph by Bertossi [7], who also authored an overview of twenty years of research in CQA [8]. Another recent overview of CQA is [37]. The term $\mathsf{CERTAINTY}(q)$ was coined in [34] to refer to CQA for Boolean queries q on databases that violate primary keys, one per relation, which are fixed by q's schema. The ICDT 2005 article of Fuxman and Miller [14, 15] started with the research of classifying all problems in the set $\{\mathsf{CERTAINTY}(q) \mid q \in \mathsf{sjfBCQ}\}$ into common complexity classes like **FO**, **P**, and **coNP**-complete. A survey of early progress in this research is [36]. A significant breakthrough was the proof that the above set exhibits an effective **P**-**coNP**-complete dichotomy [20, 21]. Furthermore, it was shown that membership of $\mathsf{CERTAINTY}(q)$ in **FO** is decidable for queries q in sjfBCQ. The current article culminates this line of research by showing that the dichotomy is actually between **L** and **coNP**-complete, and—even stronger—between expressibility in symmetric stratified Datalog (with some aggregation operator) and **coNP**-complete.

The complexity of $\mathsf{CERTAINTY}(q)$ for self-join-free conjunctive queries with negated atoms was studied in [22]. Little is known about $\mathsf{CERTAINTY}(q)$ beyond self-join-free conjunctive queries. For UCQ (i.e., unions of conjunctive queries, possibly with self-joins), Fontaine [13] showed that a **P**-**coNP**-complete dichotomy in the set $\{\mathsf{CERTAINTY}(q) \mid q$ is a Boolean query in $\mathsf{UCQ}\}$ implies Bulatov's dichotomy theorem for conservative CSP [10]. This relationship between CQA and CSP was further explored in [27]. The complexity of CQA for aggregation queries with respect to violations of functional dependencies has been studied in [3].

The counting variant of $\mathsf{CERTAINTY}(q)$, which is called $\#\mathsf{CERTAINTY}(q)$, asks to determine the number of repairs that satisfy some Boolean query q. In [29], the authors show a **FP**-**#P**-complete dichotomy in $\{\#\mathsf{CERTAINTY}(q) \mid q \in \mathsf{sjfBCQ}\}$. For conjunctive queries q with self-joins, the complexity of $\#\mathsf{CERTAINTY}(q)$ has been established for the case that all primary keys consist of a single attribute [30]. In recent years, CQA has also been studied beyond the setting of relational databases, in ontology-based knowledge bases [9, 24] and in graph databases [6].

3 Preliminaries

We assume an infinite total order $(\mathbf{dom}, \leq)$ of *constants*. We assume a set of *variables* disjoint with **dom**. If $\vec{x}$ is a sequence containing variables and constants, then $\mathsf{vars}(\vec{x})$ denotes the set of variables that occur in $\vec{x}$. A *valuation* over a set U of

variables is a total mapping θ from U to **dom**. At several places, it is implicitly understood that such a valuation θ is extended to be the identity on constants and on variables not in U. If $V \subseteq U$, then $\theta[V]$ denotes the restriction of θ to V. If θ is a valuation over a set U of variables, x is a variable (possibly $x \notin U$), and a is a constant, then $\theta_{[x \mapsto a]}$ is the valuation over $U \cup \{x\}$ such that $\theta_{[x \mapsto a]}(x) = a$ and for every variable y such that $y \neq x$, $\theta_{[x \mapsto a]}(y) = \theta(y)$.

Atoms and Key-equal Facts Each *relation name* R of arity n, $n \geq 1$, has a unique *primary key* which is a set $\{1, 2, \ldots, k\}$ where $1 \leq k \leq n$. We say that R has *signature* $[n, k]$ if R has arity n and primary key $\{1, 2, \ldots, k\}$. Elements of the primary key are called *primary-key positions*, while $k+1, k+2, \ldots, n$ are *non-primary-key positions*. For all positive integers n, k such that $1 \leq k \leq n$, we assume denumerably many relation names with signature $[n, k]$. Every relation name has a unique *mode*, which is a value in $\{\mathsf{c}, \mathsf{i}\}$. Informally, relation names of mode c will be used for consistent relations, while relations that may be inconsistent will have a relation name of mode i. We often write R^{c} to make clear that R is a relation name of mode c. Relation names of mode c will be a convenient tool in the theoretical development, but they also constitute a useful modeling primitive that can be put at the disposal of knowledge engineers [17].

If R is a relation name with signature $[n, k]$, then we call $R(s_1, \ldots, s_n)$ an *R-atom* (or simply atom), where each s_i is either a constant or a variable ($1 \leq i \leq n$). Such an atom is commonly written as $R(\underline{\vec{x}}, \vec{y})$ where the primary-key value $\vec{x} = s_1, \ldots, s_k$ is underlined and $\vec{y} = s_{k+1}, \ldots, s_n$. An *R-fact* (or simply fact) is an R-atom in which no variable occurs. Two facts $R_1(\underline{\vec{a}_1}, \vec{b}_1)$, $R_2(\underline{\vec{a}_2}, \vec{b}_2)$ are *key-equal*, denoted $R_1(\underline{\vec{a}_1}, \vec{b}_1) \sim R_2(\underline{\vec{a}_2}, \vec{b}_2)$, if $R_1 = R_2$ and $\vec{a}_1 = \vec{a}_2$.

We will use letters F, G, H for atoms. For an atom $F = R(\underline{\vec{x}}, \vec{y})$, we denote by $\mathsf{key}(F)$ the set of variables that occur in $\vec{x}$, and by $\mathsf{vars}(F)$ the set of variables that occur in F, that is, $\mathsf{key}(F) = \mathsf{vars}(\vec{x})$ and $\mathsf{vars}(F) = \mathsf{vars}(\vec{x}) \cup \mathsf{vars}(\vec{y})$. We sometimes blur the distinction between relation names and atoms. For example, if F is an atom, then the term F-fact refers to a fact with the same relation name as F.

Databases, Blocks, and Repairs A *database schema* is a finite set of relation names. All constructs that follow are defined relative to a fixed database schema. A *database* is a finite set **db** of facts using only the relation names of the schema such that for every relation name R of mode c, no two distinct R-facts of **db** are key-equal.

A *relation* of **db** is a maximal set of facts in **db** that all share the same relation name. A *block* of **db** is a maximal set of key-equal facts of **db**. A block of R-facts is also called an R-block. If A is a fact of **db**, then $\mathsf{block}(A, \mathbf{db})$ denotes the block of **db** that contains A. If $A = R(\underline{\vec{a}}, \vec{b})$, then $\mathsf{block}(A, \mathbf{db})$ is also denoted by $R(\underline{\vec{a}}, \vec{*})$. We write $R(\underline{\vec{a}}, \_)$ for a fact arbitrarily selected from the block $R(\underline{\vec{a}}, \vec{*})$. A database **db** is *consistent* if no two distinct facts of **db** are key-equal (i.e., if no block of **db** contains more than one fact). A *repair* of **db** is a maximal (with respect to set inclusion) consistent subset of **db**. We write $\mathsf{rset}(\mathbf{db})$ for the set of repairs of **db**.

Boolean Conjunctive Queries A *Boolean query* is a mapping q that associates a Boolean (true or false) to each database, such that q is closed under

isomorphism [25]. We write $\mathbf{db} \models q$ to denote that q associates true to $\mathbf{db}$, in which case $\mathbf{db}$ is said to *satisfy* q. A Boolean query q can be viewed as a decision problem that takes a database as input and asks whether $\mathbf{db}$ satisfies q. In this article, the complexity class **FO** stands for the set of Boolean queries that can be defined in first-order logic with equality and constant symbols (which are interpreted as themselves), but without other built-in predicates or function symbols.

A *Boolean conjunctive query* is a finite set $q = \{R_1(\underline{\vec{x}_1}, \vec{y}_1), \ldots, R_n(\underline{\vec{x}_n}, \vec{y}_n)\}$ of atoms, without equality or built-in predicates. We denote by $\mathsf{vars}(q)$ the set of variables that occur in q. The set q represents the first-order sentence

$$\exists u_1 \cdots \exists u_k \left(R_1(\underline{\vec{x}_1}, \vec{y}_1) \wedge \cdots \wedge R_n(\underline{\vec{x}_n}, \vec{y}_n)\right),$$

where $\{u_1, \ldots, u_k\} = \mathsf{vars}(q)$. This query q is satisfied by a database $\mathbf{db}$ if there exists a valuation θ over $\mathsf{vars}(q)$ such that for each $i \in \{1, \ldots, n\}$, there exists $R_i(\underline{\vec{a}}, \vec{b}) \in \mathbf{db}$ with $\vec{a} = \theta(\vec{x}_i)$ and $\vec{b} = \theta(\vec{y}_i)$.

We say that a Boolean conjunctive query q has a *self-join* if some relation name occurs more than once in q. If q has no self-join, then it is called *self-join-free*. We write sjfBCQ for the class of self-join-free Boolean conjunctive queries. If q is a query in sjfBCQ with an R-atom, then, by an abuse of notation, we sometimes write R to mean the R-atom of q.

Let θ be a valuation over some set U of variables. For every Boolean conjunctive query q, we write $\theta(q)$ for the query obtained from q by replacing all occurrences of each $x \in U \cap \mathsf{vars}(q)$ with $\theta(x)$; variables in $\mathsf{vars}(q) \setminus U$ remain unaffected (i.e., θ is understood to be the identity on variables not in U).

Atoms of Mode c The *mode* of an atom is the mode of its relation name (a value in $\{\mathsf{c}, \mathsf{i}\}$). If q is a query in sjfBCQ, then q^{cons} is the set of all atoms of q that are of mode c.

Functional Dependencies Let q be a Boolean conjunctive query. A *functional dependency for* q is an expression $X \rightarrow Y$ where $X, Y \subseteq \mathsf{vars}(q)$. Let $\mathcal{V}$ be a finite set of valuations over $\mathsf{vars}(q)$. We say that $\mathcal{V}$ *satisfies* $X \rightarrow Y$ if for all $\theta, \mu \in \mathcal{V}$, if $\theta[X] = \mu[X]$, then $\theta[Y] = \mu[Y]$. Let Σ be a set of functional dependencies for q. We say that $X \rightarrow Y$ is a *logical consequence of* Σ, denoted $\Sigma \models X \rightarrow Y$, if for every set $\mathcal{V}$ of valuations over $\mathsf{vars}(q)$, if $\mathcal{V}$ satisfies each functional dependency in Σ, then $\mathcal{V}$ satisfies $X \rightarrow Y$. Two sets of functional dependencies are *logically equivalent* if each functional dependency in either set is a logical consequence of the other set. Note that the foregoing conforms with standard dependency theory if variables are viewed as attributes, and valuations as tuples. As with standard functional dependencies, every set of functional dependencies for q is logically equivalent to a set of functional dependencies for q with singleton right-hand sides.

Consistent Query Answering For every Boolean query q, $\mathsf{CERTAINTY}(q)$ is the decision problem that takes as input a database $\mathbf{db}$, and asks whether every repair of $\mathbf{db}$ satisfies q.

The Genre of a Fact Let q be a query in sjfBCQ. For every fact A whose relation name occurs in q, we denote by $\mathsf{genre}_q(A)$ the (unique) atom of q that has the same

relation name as A. From here on, if **db** is a database that is given as an input to CERTAINTY(q), we will assume that each relation name of each fact in **db** also occurs in q. Therefore, for every $A \in \mathbf{db}$, $\mathsf{genre}_q(A)$ is well defined. Of course, this assumption is harmless.

Attack Graph Attack graphs were first introduced in [34] as a tool in the complexity classification of CERTAINTY(q); we recall next a slightly generalized notion of attack graph that first appeared in [20]. Let q be a query in sjfBCQ. We define $\mathcal{K}(q)$ as the following set of functional dependencies:

$$\mathcal{K}(q) := \{\mathsf{key}(F) \to \mathsf{vars}(F) \mid F \in q\}.$$

For every atom $F \in q$, we define $F^{+,q}$ as the following set of variables, depending on the mode of F:

- if F has mode i, then $F^{+,q} := \{x \in \mathsf{vars}(q) \mid \mathcal{K}(q \setminus \{F\}) \models \mathsf{key}(F) \to x\}$;
- if F has mode c, then $F^{+,q} := \{x \in \mathsf{vars}(q) \mid \mathcal{K}(q) \models \mathsf{key}(F) \to x\}$.

The *attack graph* of q is a directed graph whose vertices are the atoms of q. There is a directed edge from F to G ($F \neq G$), denoted $F \overset{q}{\rightsquigarrow} G$, if there exists a sequence

$$F_0 \overset{x_1}{\frown} F_1 \overset{x_2}{\frown} F_2 \cdots \overset{x_\ell}{\frown} F_\ell \tag{1}$$

such that $F_0 = F$, $F_\ell = G$, and for each $i \in \{1, \ldots, \ell\}$, F_i is an atom of q and x_i is a variable satisfying $x_i \in (\mathsf{vars}(F_{i-1}) \cap \mathsf{vars}(F_i)) \setminus F^{+,q}$. The sequence (1) is also called a *witness* for $F \overset{q}{\rightsquigarrow} G$. An edge $F \overset{q}{\rightsquigarrow} G$ is also called an *attack from F to G*; we also say that *F attacks G*. Informally, an attack from an atom $R(\underline{\vec{x}}, \vec{y})$ to an atom $S(\underline{\vec{u}}, \vec{w})$ indicates that, given a valuation over $\mathsf{vars}(\vec{x})$, the values for $\vec{u}$ that make the query true depend on the values chosen for $\vec{y}$.

An attack on a variable $x \in \mathsf{vars}(q)$ is defined as follows: $F \overset{q}{\rightsquigarrow} x$ if $F \overset{q \cup \{N(\underline{x})\}}{\rightsquigarrow} N(\underline{x})$ where N is a fresh relation name of signature [1, 1]. Informally, x is attacked in q if $N(\underline{x})$ has an incoming attack in the attack graph of $q \cup \{N(\underline{x})\}$.

Example 1 Consider the query $q_1 = \{R(\underline{x}, y), S(\underline{y}, z), U(\underline{y, z, w}, x), T_1(\underline{z}, w), T_2(\underline{z}, w), T^{\mathsf{c}}(\underline{z}, w)\}$. Using relation names for atoms, we have $R^{+,q_1} = \{x\}$. A witness for $R \overset{q_1}{\rightsquigarrow} U$ is $R \overset{y}{\frown} U$. The attack graph of q_1 is shown in Fig. 1.

An attack $F \overset{q}{\rightsquigarrow} G$ is *weak* if $\mathcal{K}(q) \models \mathsf{key}(F) \to \mathsf{key}(G)$; otherwise it is *strong*. A cycle in the attack graph is *strong* if at least one attack in the cycle is strong. It has been proved [21, Lemma 3.6] that if the attack graph contains a strong cycle, then it contains a strong cycle of length 2. The main result in [21] can now be stated.

Theorem 2 [21] *For every query q in* sjfBCQ*,*

- *if the attack graph of q is acyclic, then* CERTAINTY(q) *is in* **FO***;*
- *if the attack graph of q is cyclic but contains no strong cycle, then the problem* CERTAINTY(q) *is* ***L****-hard and in* **P***; and*

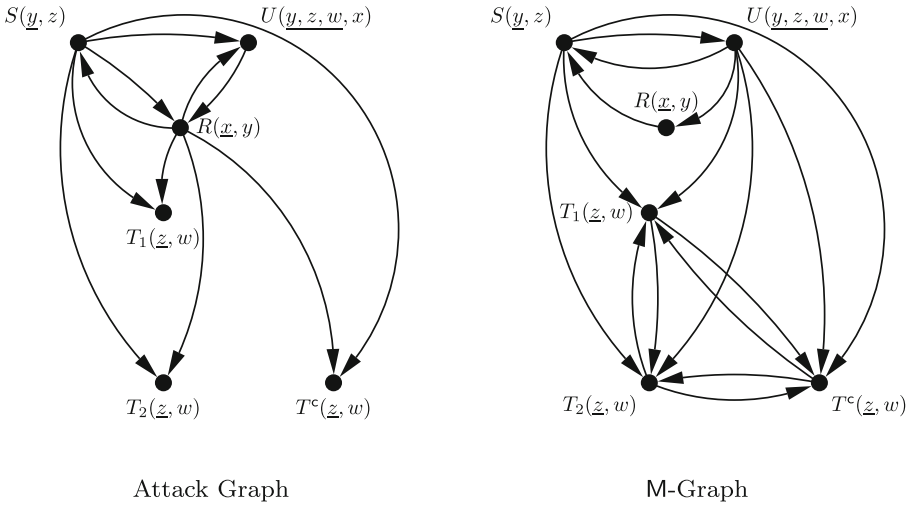


Fig. 1 Attack graph *(left)* and M-graph *(right)* of the same query $q_1 = \{R(\underline{x}, y), S(\underline{y}, z), U(\underline{y, z, w}, x), T_1(\underline{z}, w), T_2(\underline{z}, w), T^{\mathsf{c}}(\underline{z}, w)\}$. It can be verified that all attacks are weak and that the query is saturated. The attack graph has an initial strong component containing three atoms (R, S, and U). As predicted by Lemma 12, the subgraph of the M-graph induced by $\{R, S, U\}$ is cyclic

– *if the attack graph of q contains a strong cycle, then* CERTAINTY(q) *is* **coNP**-*complete.*

Furthermore, it can be decided in quadratic time in the size of q which of these three cases applies.

Reductions In this article, all considered decision problems take as input databases over some fixed schema. Let P_1 and P_2 be two such decision problems over schemas S_1 and S_2 respectively. Let $\mathcal{L}$ be some logic (e.g., first-order logic or a fragment of Datalog). A many-one reduction ρ from P_1 to P_2 is said to be *expressible in* $\mathcal{L}$ if for every n-ary relation name R in S_2, $\mathcal{L}$ contains a query $\varphi_R(x_1, \ldots, x_n)$ over S_1 such that for every database **db** over S_1, for all constants $a_1, \ldots, a_n$, we have $R(a_1, \ldots, a_n) \in \rho(\mathbf{db})$ if and only if $\mathbf{db} \models \varphi_R(a_1, \ldots, a_n)$.

Notions from Graph Theory We adopt some terminology from [4]. A directed graph is *strongly connected* if there is a directed path from any vertex to any other. The maximal strongly connected subgraphs of a graph are vertex-disjoint and are called its *strong components*. If S_1 and S_2 are strong components such that an edge leads from a vertex in S_1 to a vertex in S_2, then S_1 is a *predecessor* of S_2 and S_2 is a *successor* of S_1. A strong component is called *initial* if it has no predecessor. For a directed graph, we define the length of a directed path as the number of edges it contains. A directed path or cycle without repeated vertices is called *elementary*. If G is a graph, then $V(G)$ denotes the vertex set of G, and $E(G)$ denotes the edge set of G.

Datalog with Stratified Negation We assume that the reader is familiar with the syntax and semantics of Datalog. We fix some terminology for Datalog programs, most of which is standard. A predicate that occurs in the head of some rule is called an *intensional database predicate* (IDB predicate); otherwise it is an *extensional database predicate* (EDB predicate).

The following definition is slightly adapted from [16, p. 185]. A *stratified Datalog program* is a sequence $P = (P_0, \ldots, P_r)$ of basic Datalog programs, which are called the *strata* of P, such that each of the IDB predicates of P is an IDB predicate of precisely one stratum P_i and can be used as an EDB predicate (but not as an IDB predicate) in higher strata P_j where $j > i$. In particular, this means that

1. if an IDB predicate of stratum P_j occurs *positively* in the body of a rule of stratum P_i, then $j \leq i$, and
2. if an IDB predicate of stratum P_j occurs *negatively* in the body of a rule of stratum P_i, then $j < i$.

Stratified Datalog programs are given natural semantics using semantics for Datalog programs for each P_i, where the IDB predicates of a lower stratum are viewed as EDB predicates for a higher stratum. A rule is *recursive* if its body contains an IDB predicate of the same stratum.

Symmetric Stratified Datalog A stratified Datalog program is *linear* if in the body of each rule there is at most one occurrence of an IDB predicate of the same stratum (but there may be arbitrarily many occurrences of IDB predicates from lower strata). Assume that some stratum of a linear stratified Datalog program contains a recursive rule

$$L_0 \leftarrow L_1, L_2, \ldots, L_m, \neg L_{m+1}, \ldots, \neg L_n$$

such that L_1 is an IDB predicate of the same stratum. Then, since the program is linear, each predicate among $L_2, \ldots, L_n$ is either an EDB predicate or an IDB predicate of a lower stratum. Such a rule has a *symmetric rule*:

$$L_1 \leftarrow L_0, L_2, \ldots, L_m, \neg L_{m+1}, \ldots, \neg L_n.$$

A stratified Datalog program is *symmetric* if it is linear and the symmetric of any recursive rule is also a rule of the program.

It is known (see, for example, [16, Proposition 3.3.72]) that linear stratified Datalog is equivalent to Transitive Closure Logic. The data complexity of linear stratified Datalog is in **NL** (and is complete for **NL**). It easily follows from [12, Theorem 1] that the data complexity of symmetric stratified Datalog is in **L**. Note that all complexity results in this article refer to data complexity, i.e., complexity in the the size of the input database, for any fixed query which is not part of the input. Since we are solving a decision problem, the output can be taken to be a 0-ary predicate, for example, TrueInEveryRepair().

We will assume that given a (extensional or intensional) predicate P of some arity 2ℓ, we can express the following query (let $\vec{x} = \langle x_1, \ldots, x_\ell \rangle$, $\vec{y} = \langle y_1, \ldots, y_\ell \rangle$, and $\vec{z} = \langle z_1, \ldots, z_\ell \rangle$):

$$\{\vec{x}, \vec{y} \mid P(\vec{x}, \vec{y}) \wedge \forall z_1 \cdots \forall z_\ell \, (P(\vec{x}, \vec{z}) \rightarrow \vec{y} \leq_\ell \vec{z})\}, \tag{2}$$

where $\leq_\ell$ is a total order on $\mathbf{dom}^\ell$. Informally, the above query groups by the ℓ leftmost positions, and, within each group, takes the smallest (with respect to $\leq_\ell$) value for the remaining positions. Such a query will be useful in Section 7.4, where P encodes an equivalence relation on a finite subset of $\mathbf{dom}^\ell$, and the query (2) allows us to deterministically choose a representative in each equivalence class. The order $\leq_\ell$ can be first-order defined as the lexicographical order on $\mathbf{dom}^\ell$ induced by the linear order on **dom**. For example, for $\ell = 2$, the lexicographical order is defined as $(y_1, y_2) \leq_2 (z_1, z_2)$ if $y_1 < z_1 \vee ((y_1 = z_1) \wedge (y_2 \leq z_2))$. Nevertheless, our results do not depend on how the order $\leq_\ell$ is defined. Moreover, all queries in our study will be order-invariant in the sense defined in [18]. The order is only needed in the proof of Lemma 10 to pick, in a deterministic way, an identifier from a set of candidate identifiers. In Datalog, we use the following convenient syntax for (2):

$$\mathsf{Answer}(\vec{x}, \min(\vec{y})) \leftarrow P(\vec{x}, \vec{y}).$$

Such a rule will always be non-recursive. Most significantly, if we extend a logspace fragment of stratified Datalog with queries of the form (2), the extended fragment will also be in logspace. In particular, the query (2) can be expressed in first-order logic with a total ordering, and, consequently, is in uniform $\mathbf{AC}^0$. Therefore, assuming queries of the form (2) is harmless for our complexity-theoretic purposes. We use *SymStratDatalog* for symmetric stratified Datalog, and *SymStratDatalog*$^{\min}$ for symmetric stratified Datalog that allows queries of the form (2).

4 The Main Theorem and an Informal Guide of its Proof

In this article, we prove the following main result.

Theorem 3 (Main Theorem) *For every query q in* sjfBCQ*,*

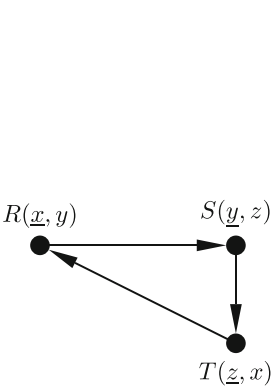
- *if the attack graph of q contains a strong cycle, then* CERTAINTY(q) *is* **coNP***-complete; and*
- *if the attack graph of q contains no strong cycle, then* CERTAINTY(q) *is expressible in SymStratDatalog*$^{\min}$ *(and therefore is in* ***L****).*

The above result is stronger than Theorem 1, since it provides an effective criterion for the dichotomy between **coNP**-completeness and expressibility in symmetric stratified Datalog.

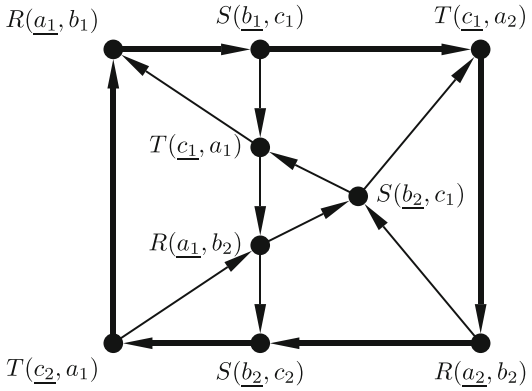
Before we delve into the proof in the next sections, we start with a guided tour that introduces our approach in an informal way. A major contribution of this article is a deterministic logspace algorithm for CERTAINTY(q) whenever CERTAINTY(q) is in **P** but not in **FO** (assuming $\mathbf{P} \neq \mathbf{coNP}$). In what follows, by a logspace algorithm, we will always mean a deterministic logspace algorithm. An exemplar query is $C_3 := \{R(\underline{x}, y), S(\underline{y}, z), T(\underline{z}, x)\}$, which can be thought of as a cycle of length 3. For the purpose of this example, let q be a query in sjfBCQ that includes C_3 as a subquery (i.e., $C_3 \subseteq q$).

An important novel notion in this article is the M-graph of a query, which will be introduced in Section 6. The M-graph of C_3 is shown in Fig. 2a. Informally, a directed edge from an atom F to an atom G, denoted $F \xrightarrow{\mathsf{M}} G$, means that every variable that occurs in the primary key of G occurs also in F. In Fig. 2a, we have $T(\underline{z}, x) \xrightarrow{\mathsf{M}} R(\underline{x}, y)$, because R's primary key (i.e., x) occurs in the T-atom; there is no edge from $R(\underline{x}, y)$ to $T(\underline{z}, x)$ because z does not occur in the R-atom. Intuitively, one can think of edges in the M-graph as foreign-to-primary key joins. In what follows, we focus on cycles in the M-graph, called M-cycles. As we will see later on, such M-cycles will occur whenever the attack graph of a query is cyclic but contains no strong attack cycles.

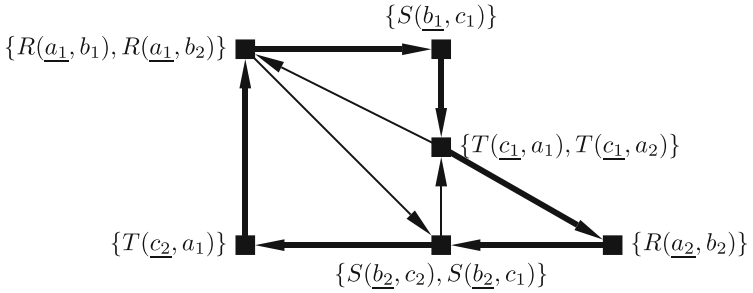
Figure 2b shows an *instantiation* of the M-graph, called $\overset{C_3}{\hookrightarrow}$-graph, whose vertices are obtained by replacing variables with constants in $R(\underline{x}, y)$, $S(\underline{y}, z)$, or $T(\underline{z}, x)$. We write $A \overset{C_3}{\hookrightarrow} B$ to denote an edge from fact A to fact B. Each triangle in the $\overset{C_3}{\hookrightarrow}$-graph



(a) M-graph.



(b) $\overset{C_3}{\hookrightarrow}$-graph.



(c) Block-quotient graph.

Fig. 2 Examples of three different graphs used in this article: M-graph, $\hookrightarrow$-graph, block-quotient graph

of Fig. 2b instantiates the query C_3; for example, the inner triangle is equal to $\theta(C_3)$ where θ is the valuation such that $\theta(xyz) = a_1b_2c_1$. We will call such a triangle a 1-embedding. Significantly, some edges are not part of any triangle. For example, the edge $S(\underline{b_1}, c_1) \overset{C_3}{\hookrightarrow} T(\underline{c_1}, a_2)$ is not in a triangle, but is present because the primary key of $T(\underline{c_1}, a_2)$ occurs in $S(\underline{b_1}, c_1)$. The notions of $\overset{C_3}{\hookrightarrow}$-graph and embedding will be defined in Definitions 4 and 5.

Let **db** be a database that is input to CERTAINTY(q) such that **db** contains (but is not limited to) all facts of Fig. 2b. Since C_3 is a subquery of q, **db** will typically contain other facts with relation names in $q \setminus C_3$. Furthermore, **db** can contain R-facts, S-facts, and T-facts not shown in Fig. 2b. Then, **db** has at least $2^3 = 8$ repairs, because Fig. 2b shows two R-facts with primary key a_1, two S-facts with primary key b_2, and two T-facts with primary key c_1. Consider now the outermost elementary cycle (in thick lines) of length 6, i.e., the cycle using the vertices in $\mathbf{r} := \{R(\underline{a_1}, b_1), S(\underline{b_1}, c_1), T(\underline{c_1}, a_2), R(\underline{a_2}, b_2), S(\underline{b_2}, c_2), T(\underline{c_2}, a_1)\}$, which will be called a 2-embedding (or an n-embedding with $n = 2$). One can verify that **r** does not contain distinct key-equal facts and does not satisfy C_3 (because the subgraph induced by **r** has no triangle). Let **o** be the database that contains **r** as well as all facts of **db** that are key-equal to some fact in **r**. A crucial observation is that if $\mathbf{db} \setminus \mathbf{o}$ has a repair that falsifies q, then so has **db** (the converse is trivially true). Indeed, if **s** is a repair of $\mathbf{db} \setminus \mathbf{o}$ that falsifies q, then $\mathbf{s} \cup \mathbf{r}$ is a repair of **db** that falsifies q. Intuitively, we can add **r** to **s** without creating a triangle in the $\overset{C_3}{\hookrightarrow}$-graph (i.e., without making C_3 true, and thus without making q true), because the facts in **r** form a cycle on their own and contain no outgoing $\overset{C_3}{\hookrightarrow}$-edges to facts in **s**. In Section 5, the set **o** will be called a *garbage set:* its facts can be thrown away without changing the answer to CERTAINTY(q). Note that the $\overset{C_3}{\hookrightarrow}$-graph of Fig. 2b contains other elementary cycles of length 6, which, however, contain distinct key-equal facts: for example, the cycle with vertices $R(\underline{a_1}, b_1), S(\underline{b_1}, c_1), T(\underline{c_1}, a_1), R(\underline{a_1}, b_2), S(\underline{b_2}, c_2), T(\underline{c_2}, a_1)$ contains both $R(\underline{a_1}, b_1)$ and $R(\underline{a_1}, b_2)$.

Garbage sets thus arise from cycles in the $\overset{C_3}{\hookrightarrow}$-graph that *(i)* do not contain distinct key-equal facts, and *(ii)* are not triangles satisfying C_3. To find such cycles, we construct the quotient graph of the $\overset{C_3}{\hookrightarrow}$-graph with respect to the equivalence relation "is key-equal to." Since the equivalence classes with respect to "is key-equal to" are the *blocks* of the database, we call this graph the *block-quotient graph* (Definition 6). The block-quotient graph for our example is shown in Fig. 2c. The vertices are database blocks; there is an edge from block $\mathbf{b}_1$ to $\mathbf{b}_2$ if the $\overset{C_3}{\hookrightarrow}$-graph contains an edge from some fact in $\mathbf{b}_1$ to some fact in $\mathbf{b}_2$. The block-quotient graph contains exactly one elementary directed cycle of length 6 (thick lines); this cycle obviously corresponds to the outermost cycle of length 6 in the $\overset{C_3}{\hookrightarrow}$-graph. A core result (Lemma 8) of this article is a logspace algorithm for finding elementary cycles in the block-quotient graph whose lengths are strict multiples of the length of the underlying M-cycle. In our example, since the M-cycle of C_3 has length 3, we are looking for cycles in the block-quotient graph of lengths $6, 9, 12, \ldots$ Note here that, since the $\overset{C_3}{\hookrightarrow}$-graph is

tripartite, the length of any cycle in it must be a multiple of 3. Our algorithm can be encoded in symmetric stratified Datalog. This core algorithm is then extended, in the proof of Lemma 9, to compute garbage sets for M-cycles.

In our example, C_3 is a subquery of q. In general, M-cycles will be subqueries of larger queries. The facts that belong to the garbage set for an M-cycle can be removed, but the other facts must be maintained for computations on the remaining part of the query, and are stored in a new schema that replaces the relations in the M-cycle with a single relation (see Section 7.4). In our example, this new relation has attributes for x, y, and z, and stores all triangles that are outside the garbage set for C_3.

We can now sketch our approach for dealing with queries q such that CERTAINTY(q) is in $\mathbf{P}\setminus\mathbf{FO}$. Lemma 12 will tell us that such a query q will have an M-cycle involving two or more atoms of mode i. The garbage set of this M-cycle is then computed, and the facts not in the garbage set will be stored in a single new relation of mode i that replaces the M-cycle. In this way, CERTAINTY(q) is reduced to a new problem CERTAINTY(q'), where q' contains less atoms of mode i than q. Lemma 10 shows that this new problem will be in $\mathbf{P}$, and that our reduction can be expressed in symmetric stratified Datalog. We can repeat this reduction until we arrive at a query q'' such that CERTAINTY(q'') is in $\mathbf{FO}$.

To conclude this guided tour, we point out the role of atoms of mode c in the computation of the M-graph, which was not illustrated by our running example. In the M-graph of Fig. 1 *(right)*, we have $S(\underline{y}, z) \xrightarrow{\mathsf{M}} U(\underline{y, z}, w, x)$, even though w does not occur in the S-atom. The explanation is that the query also contains the consistent relation $T^{\mathsf{c}}(\underline{z}, w)$, which maps each z-value to a unique w-value. So even though w does not occur in $S(\underline{y}, z)$, it is nevertheless uniquely determined by z. It is therefore important to identify all relations of mode c, which is the topic of Section 8.1.

5 Garbage Sets

Let **db** be a database that is an input to CERTAINTY(q) with $q \in$ sjfBCQ. In this section, we show that it is generally possible to downsize **db** by deleting blocks from it without changing the answer to CERTAINTY(q). That is, if the downsized database has a repair falsifying q, then so does the original database (the converse holds trivially true). Intuitively, the deleted blocks can be considered as "garbage" for the problem CERTAINTY(q).

Definition 2 The following definition is relative to a fixed query q in sjfBCQ. Let $q_0 \subseteq q$. Let **db** be a database. We say that a subset **o** of **db** is a *garbage set for q_0 in* **db** if the following conditions are satisfied:

1. for every $A \in \mathbf{o}$, we have that $\mathsf{genre}_q(A) \in q_0$ and $\mathsf{block}(A, \mathbf{db}) \subseteq \mathbf{o}$; and
2. there exists a repair **r** of **o** such that for every valuation θ over $\mathsf{vars}(q)$, if $\theta(q) \subseteq (\mathbf{db} \setminus \mathbf{o}) \cup \mathbf{r}$, then $\theta(q_0) \cap \mathbf{r} = \emptyset$ (and therefore $\theta(q_0) \subseteq \mathbf{db} \setminus \mathbf{o}$).

The first condition in the above definition says that the relation names of facts in **o** must occur in q_0, and that every block of **db** is either included in or disjoint with

o. The second condition captures the crux of the definition and was illustrated in Section 4.

We now show a number of useful properties of garbage sets that are quite intuitive. In particular, by Lemma 1, there exists a unique maximal (with respect to $\subseteq$) garbage set for q_0 in **db**, which will be called *the maximum garbage set for* q_0 *in* **db**.

Lemma 1 *Let q be a query in* sjfBCQ*, and let $q_0 \subseteq q$. Let* **db** *be a database. If* $\mathbf{o}_1$ *and* $\mathbf{o}_2$ *are garbage sets for q_0 in* **db***, then* $\mathbf{o}_1 \cup \mathbf{o}_2$ *is a garbage set for q_0 in* **db***.*

Lemma 2 *Let q be a query in* sjfBCQ*, and let $q_0 \subseteq q$. Let* **db** *be a database. Let* **o** *be a garbage set for q_0 in* **db***. Then, every repair of* **db** *satisfies q if and only if every repair of* **db** $\setminus$ **o** *satisfies q (i.e.,* **db** *and* **db** $\setminus$ **o** *agree on their answer to* CERTAINTY(q)*).*

Lemma 3 *Let q be a query in* sjfBCQ*, and let $q_0 \subseteq q$. Let* **db** *be a database. Let* **o** *be a garbage set for q_0 in* **db***. Then, every garbage set for q_0 in* **db** $\setminus$ **o** *is empty if and only if* **o** *is the maximum garbage set for q_0 in* **db***.*

The next two sections focus on the computation of garbage sets for subqueries of a particular type. These subqueries correspond to cycles in the M-graph of a query, which is defined next.

6 M-Graphs and $\rightsquigarrow$-Graphs

In this section, we introduce the M-*graph* of a query q in sjfBCQ, which is a generalization of the notion of Markov-graph introduced in [21] (hence the use of the letter M). We also define $\rightsquigarrow$-graphs, which can be regarded as data-level instantiations of M-graphs.

Definition 3 Let q be a query in sjfBCQ. The M-*graph* of q is a directed graph whose vertices are the atoms of q. There is a directed edge from F to G ($F \neq G$), denoted $F \xrightarrow{\mathsf{M}} G$, if $\mathcal{K}(q^{\mathsf{cons}}) \models \mathsf{vars}(F) \rightarrow \mathsf{key}(G)$. A cycle in the M-graph is called an M-*cycle*.

Note that if all relation names in q have mode i, then $F \xrightarrow{\mathsf{M}} G$ implies $\mathsf{key}(G) \subseteq \mathsf{vars}(F)$. M-Graphs are technically easier to deal with than Markov-graphs [21] on which they are inspired. In fact, Markov-graphs in [21] were only defined for queries not containing any atom of mode i with a composite primary key. Therefore, atoms with composite primary keys had first to be massaged into the form required by Markov-graphs. This drawback is resolved by the new notion of M-graph.

Example 2 The notion of M-graph is illustrated by Fig. 1. We have $\mathcal{K}(q_1{}^{\mathsf{cons}}) = \{z \rightarrow w\}$. Since $\mathcal{K}(q_1{}^{\mathsf{cons}}) \models \mathsf{vars}(S) \rightarrow \mathsf{key}(U)$, the M-graph has a directed edge from S to U.

Given a query q, every database that instantiates the schema of q naturally gives rise to an instantiation of the $\xrightarrow{\mathsf{M}}$-edges in q's M-graph, in a way that is captured by the following definition.

Definition 4 The following notions are defined relative to a query q in sjfBCQ and a database **db**. The $\hookrightarrow$*-graph* of **db** is a directed graph whose vertices are the atoms of **db**. There is a directed edge from A to B, denoted $A \hookrightarrow B$, if there exists a valuation θ over $\mathsf{vars}(q)$ and an edge $F \xrightarrow{\mathsf{M}} G$ in the M-graph of q such that $\theta(q) \subseteq \mathbf{db}$, $A = \theta(F)$, and $B \sim \theta(G)$. A cycle in the $\hookrightarrow$-graph is also called a $\hookrightarrow$-cycle. Informally, one can think of the $\hookrightarrow$-graph as the *instantiated* M*-graph*.

The notion of $\hookrightarrow$-graph is illustrated by Fig. 3. The following lemma states that if the $\hookrightarrow$-graph of a database **db** has a directed edge from some fact A to some G-fact B, then A has outgoing edges to all the facts of $\mathsf{block}(B, \mathbf{db})$, and to no other G-facts.

Lemma 4 *Let* $q \in$ sjfBCQ *and let* **db** *be a database. Then,*

1. *for all* $A, B \in \mathbf{db}$*, if* $A \hookrightarrow B$*, then* $A \hookrightarrow B'$ *for all* $B' \in \mathsf{block}(B, \mathbf{db})$*;*
2. *for all* $A, B, B' \in \mathbf{db}$*, if* $A \hookrightarrow B$ *and* $A \hookrightarrow B'$ *and* $\mathsf{genre}_q(B) = \mathsf{genre}_q(B')$*, then* $B \sim B'$*.*

Proof The first item is trivial. For the second item, assume $A \hookrightarrow B$, $A \hookrightarrow B'$, and $\mathsf{genre}_q(B) = \mathsf{genre}_q(B')$. We can assume $F, G \in q$ such that $F \xrightarrow{\mathsf{M}} G$, $\mathsf{genre}_q(A) = F$, and $\mathsf{genre}_q(B) = G$. Then, there exist valuations θ_1, θ_2 over $\mathsf{vars}(q)$ such that $A \in \theta_1(q) \subseteq \mathbf{db}$, $A \in \theta_2(q) \subseteq \mathbf{db}$, $B \sim \theta_1(G)$, and $B' \sim \theta_2(G)$. Since $\theta_1[\mathsf{vars}(F)] = \theta_2[\mathsf{vars}(F)]$ and $\mathcal{K}(q^{\mathsf{cons}}) \models \mathsf{vars}(F) \to \mathsf{key}(G)$ (because $F \xrightarrow{\mathsf{M}} G$), it follows $\theta_1[\mathsf{key}(G)] = \theta_2[\mathsf{key}(G)]$, hence B and B' must be key-equal. □

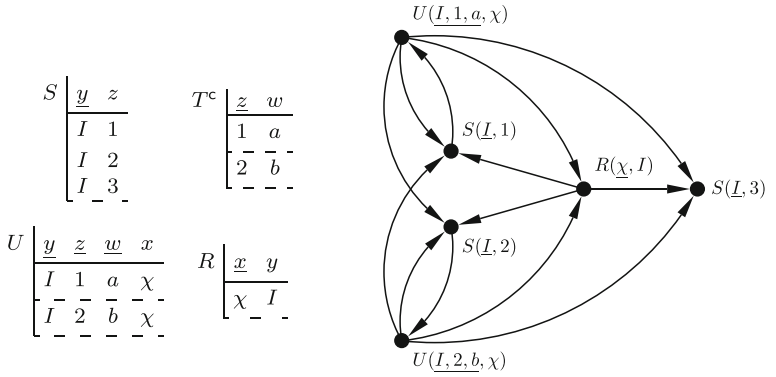


Fig. 3 *Left:* Database that is input to CERTAINTY(q_1) for the query q_1 in Fig. 1. The relations for T_1 and T_2, which are identical to the relation for T^{c}, have been omitted. *Right:* The $\hookrightarrow$-graph from which, for readability reasons, T_1-facts, T_2-facts, and T^{c}-facts have been omitted

7 Garbage Sets for M-Cycles

In this section, we bring together notions of the two preceding sections. We focus on queries q in sjfBCQ whose M-graph has a cycle C. From here on, if C is an elementary cycle in the M-graph of some query q in sjfBCQ, then the subset of q that contains all (and only) the atoms of C, is also denoted by C.

Section 7.1 shows a procedural characterization of the maximum garbage set for C. Section 7.2 shows that this maximum garbage set can be computed in linear stratified Datalog, and Section 7.3 shows that the computation is even possible in symmetric stratified Datalog. Finally, Section 7.4 shows a reduction, also expressible in symmetric stratified Datalog, that replaces C with a single atom. The elimination of M-cycles is crucial in the proof of Theorem 3, which will be given in Section 8. In particular, it will be shown there that if CERTAINTY(q) is in $\mathbf{P} \setminus \mathbf{FO}$, then q's M-graph has a cycle that can be eliminated.

7.1 Characterizing Garbage Sets for M-Cycles

We define how a given M-cycle C of length k can be instantiated by cycles in the $\rightsquigarrow$-graph, called *embeddings*, whose lengths are multiples of k.

Definition 5 Let q be a query in sjfBCQ. Let **db** be a database. Let C be an elementary directed cycle in the M-graph of q. The cycle C naturally induces a subgraph of the $\rightsquigarrow$-graph, as follows: the vertex set of the subgraph contains all (and only) the facts A of **db** such that $\mathsf{genre}_q(A)$ is an atom in C; there is a directed edge from A to B, denoted $A \overset{C}{\rightsquigarrow} B$, if $A \rightsquigarrow B$ and the cycle C contains a directed edge from $\mathsf{genre}_q(A)$ to $\mathsf{genre}_q(B)$.

Let k be the length of C. Obviously, the length of every $\overset{C}{\rightsquigarrow}$-cycle must be a multiple of k. Let n be a positive integer. An *n-embedding of C in* **db** (or simply *embedding* if the value n is not important) is an elementary $\overset{C}{\rightsquigarrow}$-cycle of length nk containing no two distinct key-equal facts. A 1-embedding of C in **db** is said to be *relevant* if there exists a valuation θ over $\mathsf{vars}(q)$ such that $\theta(q) \subseteq$ **db** and $\theta(q)$ contains every fact of the 1-embedding; otherwise the 1-embedding is said to be *irrelevant*.

Let C and q be as in Definition 5, and let **db** be a database. We next point out an intimate relationship between garbage sets for C in **db** and different sorts of embeddings.

- Let $A \in$ **db** such that $\mathsf{genre}_q(A)$ belongs to C. If A belongs to some relevant 1-embedding of C in **db**, then A will have an outgoing edge in the $\overset{C}{\rightsquigarrow}$-graph. If A does not belong to some relevant 1-embedding of C in **db**, then A will have no outgoing edge in the $\overset{C}{\rightsquigarrow}$-graph, and $\mathsf{block}(A, \mathbf{db})$ is a garbage set for C in **db** by Definition 2 (choose $\mathbf{o} = \mathsf{block}(A, \mathbf{db})$ and $\mathbf{r} = \{A\}$). Note incidentally that if A belongs to an irrelevant 1-embedding, then A must have an outgoing edge in the $\overset{C}{\rightsquigarrow}$-graph, which implies that A also belongs to a relevant 1-embedding.

- Every irrelevant 1-embedding of C in **db** gives rise to a garbage set. To illustrate this case, let $C = \{R(\underline{x}, y, z),\ S(\underline{y}, x, z)\}$. Assume that $R(\underline{a}, b, 1) \overset{C}{\hookrightarrow} S(\underline{b}, a, 2) \overset{C}{\hookrightarrow} R(\underline{a}, b, 1)$ is a 1-embedding of C in **db**. This 1-embedding is irrelevant, because $1 \neq 2$, whereas the third positions of R and S in C are equal to z. It can be easily seen that $R(\underline{a}, *, *) \cup S(\underline{b}, *, *)$ is a garbage set for q in **db**, where q is the query that contains C.
- Every n-embedding of C in **db** with $n \geq 2$ gives rise to a garbage set. This was illustrated in Section 4 by means of the outermost cycle of length 6 in Fig. 2b, which is a 2-embedding of $\{R(\underline{x}, y),\ S(\underline{y}, z),\ T(\underline{z}, x)\}$.

These observations lead to the following lemma which provides a procedural characterization of the maximum garbage set for C in a given database.

Lemma 5 *Let q be a query in* sjfBCQ. *Let $C = F_0 \xrightarrow{\mathsf{M}} F_1 \xrightarrow{\mathsf{M}} \cdots \xrightarrow{\mathsf{M}} F_{k-1} \xrightarrow{\mathsf{M}} F_0$ be an elementary cycle of length k ($k \geq 2$) in the* M*-graph of q. Let* **db** *be a database. Let* **o** *be a minimal (with respect to $\subseteq$) subset of* **db** *satisfying the following conditions:*

1. *the set* **o** *contains every fact A of* **db** *with* $\mathsf{genre}_q(A) \in \{F_0, \ldots, F_{k-1}\}$ *such that A has zero outdegree in the $\overset{C}{\hookrightarrow}$-graph;*
2. *the set* **o** *contains every fact that belongs to some irrelevant 1-embedding of C in* **db***;*
3. *the set* **o** *contains every fact that belongs to some n-embedding of C in* **db** *with $n \geq 2$;*
4. *Recursive condition: if* **o** *contains some fact of a relevant 1-embedding of C in* **db***, then* **o** *contains every fact of that 1-embedding; and*
5. *Closure under "is key-equal to": if* **o** *contains some fact A, then* **o** *includes* $\mathsf{block}(A, \mathbf{db})$.

Then, **o** *is the maximum garbage set for C in* **db**.

7.2 Computing n-Embeddings of M-Cycles, $n \geq 2$

In this and the following section, we translate Lemma 5 into a Datalog program that computes the maximum garbage set for an M-cycle C. The main computational challenge lies in Condition 3 of Lemma 5, which adds to the maximum garbage set all facts belonging to some n-embedding with $n \geq 2$, where the value of n is not upper bounded. Such n-embeddings can be computed in nondeterministic logspace by using directed reachability. In this section, we express this computation in linear stratified Datalog. In the following section, we show that the computation is even possible in symmetric stratified Datalog.

We will compute reachability in the quotient graph of the $\overset{C}{\hookrightarrow}$-graph relative to the equivalence relation "is key-equal to," as defined next.

Definition 6 Let q be a query in sjfBCQ. Let **db** be a database. Let C be an elementary directed cycle of length $k \geq 2$ in the M-graph of q. The *block-quotient*

graph is the quotient graph of the $\overset{C}{\hookrightarrow}$-graph of **db** with respect to the equivalence relation $\sim$. [1]

The edge set of the block-quotient graph of a database can obviously be computed in **FO**. The following lemma states a correspondence between n-embeddings and elementary cycles in the block-quotient graph.

Lemma 6 *Let q be a query in* sjfBCQ. *Let* **db** *be a database. Let C be an elementary directed cycle of length $k \geq 2$ in the* M*-graph of q. The following are equivalent for every positive integer n:*

1. *if $A_0 \overset{C}{\hookrightarrow} A_1 \overset{C}{\hookrightarrow} \cdots \overset{C}{\hookrightarrow} A_{nk-1} \overset{C}{\hookrightarrow} A_0$ is an n-embedding of C in* **db** *and $\mathbf{b}_i := \mathsf{block}(A_i, \mathbf{db})$ for $0 \leq i \leq nk-1$, then $(\mathbf{b}_0, \mathbf{b}_1, \ldots, \mathbf{b}_{nk-1}, \mathbf{b}_0)$ is an elementary directed cycle in the block-quotient graph; and*
2. *whenever $(\mathbf{b}_0, \mathbf{b}_1, \ldots, \mathbf{b}_{nk-1}, \mathbf{b}_0)$ is an elementary directed cycle in the block-quotient graph, then there exist $A_0 \in \mathbf{b}_0, \ldots, A_{nk-1} \in \mathbf{b}_{nk-1}$ such that $A_0 \overset{C}{\hookrightarrow} A_1 \overset{C}{\hookrightarrow} \cdots \overset{C}{\hookrightarrow} A_{nk-1} \overset{C}{\hookrightarrow} A_0$ is an n-embedding of C in* **db**.

Lemma 6 tells us that there is an n-embedding of C in **db**, for some $n \geq 2$, if and only if the block-quotient graph has an elementary directed cycle whose length is a strict multiple of k, where k is the length of C. To test whether such cycles exist in the block-quotient graph, we will verify the existence of a directed open path $P_{forth} := (\mathbf{b}_0, \mathbf{b}_1, \ldots, \mathbf{b}_{k-1}, \mathbf{b}_k)$, with k edges, for which there is a directed path P_{back} from $\mathbf{b}_k$ to $\mathbf{b}_0$ that uses no vertices among $\mathbf{b}_1, \ldots, \mathbf{b}_{k-1}$. In Datalog, we will define a $(k+1)$-ary predicate Pk for P_{forth}, and a $(k+1)$-ary predicate DCon for P_{back}. In the remainder of this section, we give a full program in linear stratified Datalog, and thus in **NL**. But recall that this program will be replaced, in the next subsection, by a program in symmetric stratified Datalog, with data complexity in **L**. We still give this theoretically suboptimal program because of its simplicity and because the theoretical optimization from **NL** to **L** may not be significant in practice when existing Datalog engines are used.

From here on, let the elementary cycle in the M-graph be $C = F_0 \overset{\mathsf{M}}{\longrightarrow} F_1 \overset{\mathsf{M}}{\longrightarrow} \cdots \overset{\mathsf{M}}{\longrightarrow} F_{k-1} \overset{\mathsf{M}}{\longrightarrow} F_0$ where $k \geq 2$ is the length of the cycle. For each $i \in \{0, \ldots, k-1\}$, let $F_i = R_i(\underline{\vec{x}_i}, \vec{y}_i)$. For every variable x and every $i \in \{\dagger, \ddagger\} \cup \{0, 1, 2, \ldots\}$, we write $x^{(i)}$ to denote a fresh variable such that $x^{(i)} = y^{(j)}$ if and only if $x = y$ and $i = j$. This notation extends to sequences of variables and queries in the natural way. For example, if $\vec{x} = \langle x_1, x_2, \ldots, x_n \rangle$, then $\vec{x}^{(i)} = \langle {x_1}^{(i)}, {x_2}^{(i)}, \ldots, {x_n}^{(i)} \rangle$. If c is a constant, then we define $c^{(i)} = c$. If q is a query, then $q^{(i)}$ is the query obtained from q by replacing all occurrences of every variable x with $x^{(i)}$.

[1] The quotient graph of a directed graph $G = (V, E)$ with respect to an equivalence relation $\equiv$ on V is a directed graph whose vertices are the equivalence classes of $\equiv$; there is a directed edge from class A to class B if E has a directed edge from some vertex in A to some vertex in B.

The following Datalog rule for the IDB predicate Pk computes open paths in the block-quotient graph with k edges (and $k+1$ vertices): the IDB fact $\mathsf{Pk}(\vec{a}_0, \vec{a}_1, \ldots, \vec{a}_{k-1}, \vec{a}_k)$ holds true if $\vec{a}_0 \neq \vec{a}_k$ and the block-quotient graph has directed edges going from $R_i(\underline{\vec{a}_i}, *)$ to $R_{i+1}(\underline{\vec{a}_{i+1}}, *)$ for $0 \leq i \leq k-1$. The predicates $=_{R_i}$ and $\neq_{R_i}$ test, respectively, for equality and disequality of primary-key values of R_i-facts; these predicates can be easily expressed in linear stratified Datalog (for details, see the proof of of Lemma 9).

$$\mathsf{Pk}(\vec{x}_0^{(0)}, \vec{x}_1^{(1)}, \ldots, \vec{x}_{k-1}^{(k-1)}, \vec{x}_0^{(k)}) \leftarrow \left\{ \begin{array}{ll} q^{(0)}, q^{(1)}, \ldots, q^{(k-1)}, q^{(k)}, & \\ \vec{x}_1^{(1)} & =_{R_1} \vec{x}_1^{(0)}, \\ \vec{x}_2^{(2)} & =_{R_2} \vec{x}_2^{(1)}, \\ & \vdots \\ \vec{x}_{k-1}^{(k-1)} & =_{R_{k-1}} \vec{x}_{k-1}^{(k-2)}, \\ \vec{x}_0^{(k)} & =_{R_0} \vec{x}_0^{(k-1)}, \\ \vec{x}_0^{(0)} & \neq_{R_0} \vec{x}_0^{(k)} \end{array} \right. \tag{3}$$

The rule is visualized by Fig. 4 and illustrated by Example 3 for $k = 3$, for a fact $\mathsf{Pk}(\vec{a}_0, \vec{a}_1, \vec{a}_2, \vec{a}_3)$. The four triangles $q^{(0)}, \ldots, q^{(3)}$ in Fig. 4 are relevant 1-embeddings. Every fact $R_i(\underline{\vec{a}_i}, \sqcup)$ has two outgoing edges, pointing to key-equal database facts in different embeddings. For instance, $R_0(\underline{\vec{a}_0}, \sqcup)$ has outgoing edges to two open-square vertices, which represent key-equal database facts. If we require $\vec{a}_0$ and $\vec{a}_3$ to be distinct, then the curved edges form an open path.

Example 3 Let $q = \{R_0(\underline{x}, y), R_1(\underline{y}, z), R_2(\underline{z}, x)\}$. Then, for $i \in \{0, 1, 2, 3\}$, we obtain $q^{(i)}$ by making a copy of q with all variables renamed. In this example, we

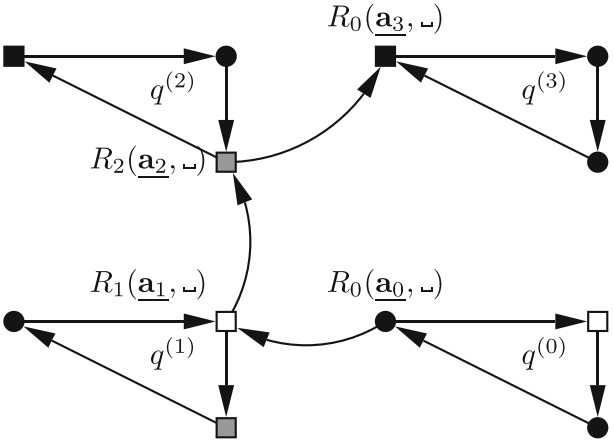


Fig. 4 Illustration of the $\hookrightarrow$-subgraph searched for by the body of the rule for Pk with $k = 3$. Rectangular nodes with the same color (white, gray, black) are key-equal. The curved arrows form a path of length k

use the renaming $\{x \mapsto x_i,\ y \mapsto y_i,\ z \mapsto z_i\}$. Then, rule (3) is as follows (up to a renaming of variables):

$$\mathsf{Pk}(x_0, y_1, z_2, x_3) \leftarrow \begin{cases} R_0(\underline{x_0}, y_0),\ R_1(\underline{y_0}, z_0),\ R_2(\underline{z_0}, x_0), \\ R_0(\underline{x_1}, y_1),\ R_1(\underline{y_1}, z_1),\ R_2(\underline{z_1}, x_1), \\ R_0(\underline{x_2}, y_2),\ R_1(\underline{y_2}, z_2),\ R_2(\underline{z_2}, x_2), \\ R_0(\underline{x_3}, y_3),\ R_1(\underline{y_3}, z_3),\ R_2(\underline{z_3}, x_3), \\ y_1 = y_0, \\ z_2 = z_1, \\ x_3 = x_2, \\ x_0 \neq x_3 \end{cases}$$

Note incidentally that the latter rule can be equivalently shortened as follows, by identifying x_3 and x_2.

$$\mathsf{Pk}(x_0, y_1, z_2, x_2) \leftarrow \begin{cases} R_0(\underline{x_0}, y_0),\ R_1(\underline{y_0}, z_0),\ R_2(\underline{z_0}, x_0), \\ R_0(\underline{x_1}, y_1),\ R_1(\underline{y_1}, z_1),\ R_2(\underline{z_1}, x_1), \\ R_0(\underline{x_2}, y_2),\ R_1(\underline{y_2}, z_2),\ R_2(\underline{z_2}, x_2), \\ y_1 = y_0, \\ z_2 = z_1, \\ x_0 \neq x_2 \end{cases}$$

This simplification can be generalized and applied to (3). However, such simplifications are beyond the focus of the current paper.

The following rules use directed reachability in the block-quotient graph. A fact $\mathsf{DCon}(\vec{a}_0, \vec{a}_0^{\dagger}, \vec{a}_1, \ldots, \vec{a}_{k-1})$ is true if $\mathsf{Pk}(\vec{a}_0', \vec{a}_1, \ldots, \vec{a}_{k-1}, \vec{a}_0'')$ is true, for some $\vec{a}_0'$ and $\vec{a}_0''$, and the block-quotient graph has a directed path from $R_0(\underline{\vec{a}_0}, *)$ to $R_0(\underline{\vec{a}_0^{\dagger}}, *)$ that uses no vertices in $\{R_i(\underline{\vec{a}_i}, *)\}_{i=1}^{k-1}$. The recursive rule for DCon is:

$$\mathsf{DCon}(\vec{x}_0^{(0)}, \vec{x}_0^{(\dagger)}, \vec{x}_1^{(1)}, \vec{x}_2^{(2)}, \ldots, \vec{x}_{k-1}^{(k-1)}) \leftarrow \begin{cases} \mathsf{DCon}(\vec{x}_0^{(0)}, \vec{x}_0^{(\ddagger)}, \vec{x}_1^{(1)}, \vec{x}_2^{(2)}, \ldots, \vec{x}_{k-1}^{(k-1)}), \\ \mathsf{Pk}(\vec{x}_0^{(\ddagger)}, \vec{x}_1^{(k+1)}, \vec{x}_2^{(k+2)}, \ldots, \vec{x}_{k-1}^{(2k-1)}, \vec{x}_0^{(\dagger)}), \\ \vec{x}_1^{(k+1)} \neq_{R_1} \vec{x}_1^{(1)}, \\ \vec{x}_2^{(k+2)} \neq_{R_2} \vec{x}_2^{(2)}, \\ \vdots \\ \vec{x}_{k-1}^{(2k-1)} \neq_{R_{k-1}} \vec{x}_{k-1}^{(k-1)} \end{cases}$$

The non-recursive rule for DCon is:

$$\mathsf{DCon}(\vec{x}_0^{(\dagger)}, \vec{x}_0^{(\dagger)}, \vec{x}_1^{(1)}, \ldots, \vec{x}_{k-1}^{(k-1)}) \leftarrow q^{(\dagger)},\ \mathsf{Pk}(\vec{x}_0^{(0)}, \vec{x}_1^{(1)}, \ldots, \vec{x}_{k-1}^{(k-1)}, \vec{x}_0^{(k)})$$

Finally, an n-embedding exists whenever we have $\mathsf{Pk}(\vec{a}_0, \vec{a}_1, \ldots, \vec{a}_{k-1}, \vec{a}_k)$ together with a directed path in the block-quotient graph from $R_0(\underline{\vec{a}_k}, *)$ to $R_0(\underline{\vec{a}_0}, *)$ that uses no vertices in $\{R_i(\underline{\vec{a}_i}, *)\}_{i=1}^{k-1}$.

$$\mathsf{InLongDCycle}(\vec{x}_0^{(0)}, \vec{x}_1^{(1)}, \ldots, \vec{x}_{k-1}^{(k-1)}) \leftarrow \begin{cases} \mathsf{Pk}(\vec{x}_0^{(0)}, \vec{x}_1^{(1)}, \ldots, \vec{x}_{k-1}^{(k-1)}, \vec{x}_0^{(k)}), \\ \mathsf{DCon}(\vec{x}_0^{(k)}, \vec{x}_0^{(0)}, \vec{x}_1^{(1)}, \ldots, \vec{x}_{k-1}^{(k-1)}) \end{cases}$$

7.3 Computing Garbage Sets for M-Cycles

In Section 7.2, n-embeddings were computed in nondeterministic logspace by using directed reachability in the block-quotient graph. This section shows a trick that allows doing the computation by using only *undirected* reachability, which, by the use of Reingold's algorithm [33], will lead to an algorithm that runs in deterministic logspace and can be expressed in *SymStratDatalog*.

The following Lemma 7 implies that if a given vertex of the $\overset{C}{\hookrightarrow}$-graph is in a strong component that has an n-embedding with $n \geq 2$, then that vertex is a database fact of the maximum garbage set. Consequently, for the purpose of constructing the maximum garbage set, it suffices to solve the decision problem that asks whether a given strong component of the $\overset{C}{\hookrightarrow}$-graph has an n-embedding with $n \geq 2$. The latter problem can be first-order reduced to a problem called $\mathsf{LONGCYCLE}(k)$ whose complexity will be established by Lemma 8.

Lemma 7 *Let C be an elementary cycle in the* M*-graph of a query q in* sjfBCQ*. Let $\mathcal{S}$ be a strong component in the $\overset{C}{\hookrightarrow}$-graph of a database* **db***. If some fact of $\mathcal{S}$ belongs to the maximum garbage set for C in* **db***, then every fact of $\mathcal{S}$ belongs to the maximum garbage set for C in* **db***.*

Proof We first show the following property: if $A \overset{C}{\hookrightarrow} B$ and B belongs to the maximum garbage set for C in **db**, then A also belongs to the maximum garbage set for C in **db**. To this end, assume $A \overset{C}{\hookrightarrow} B$ such that B belongs to the maximum garbage set for C in **db**. We can assume an edge $F_0 \xrightarrow{\mathsf{M}} F_1$ in C and a valuation θ over $\mathsf{vars}(q)$ such that $\theta(q) \subseteq \mathbf{db}$, $A = \theta(F_0)$, and $B \sim \theta(F_1)$. Since B belongs to the maximum garbage set for C in **db**, we have that $\theta(F_1)$ belongs to the maximum garbage set by Definition 2. From $\theta(C) \subseteq \mathbf{db}$, it follows that $A \overset{C}{\hookrightarrow} \theta(F_1)$ is an edge of a relevant 1-embedding of C in **db**. It follows from the recursive Condition 4 in Lemma 5 that A belongs to the maximum garbage set for C in **db**.

The proof of the lemma can now be given. Assume that $B \in \mathcal{S}$ belongs to the maximum garbage set for C in **db**. Let $A \in \mathcal{S}$. Since $\mathcal{S}$ is a strong component, there exists a path $A_0 \overset{C}{\hookrightarrow} A_1 \overset{C}{\hookrightarrow} \cdots \overset{C}{\hookrightarrow} A_\ell$ such that $A_0 = A$ and $A_\ell = B$. By repeating the property of the previous paragraph, we find that the maximum garbage set for C in **db** contains $A_\ell, A_{\ell-1}, \ldots, A_0$. □

Definition 7 Let k be an integer such that $k \geq 2$. A *k-circle-layered graph* [26] is a k-partite directed graph $G = (V, E)$ where edges only exist between adjacent partitions. More formally, the vertices of G can be partitioned into k groups such that $V = V_0 \cup V_1 \cup \cdots \cup V_{k-1}$ and $V_i \cap V_j = \emptyset$ if $i \neq j$. The only edges from a partition V_i go to the partition $V_{(i+1) \mod k}$. The problem LONGCYCLE(k) is now defined as follows:

Problem: LONGCYCLE(k)
Instance: A connected k-circle-layered graph $G = (V, E)$ such that every edge of E belongs to a directed cycle of length k.
Question: Does G have an elementary directed cycle of length at least $2k$?

Lemma 8 *For every integer k such that $k \geq 2$, LONGCYCLE(k) can be expressed in SymStratDatalog (and therefore is in **L**).*

Sketch Let $G = (V, E)$ be an instance of LONGCYCLE(k). A cycle of length k in G is called a *k-cycle*. Let $\widehat{G}$ be the undirected graph whose vertices are the k-cycles of G; there is an undirected edge between two vertices if their k-cycles have an element in common. The full proof in Appendix C shows that G has an elementary directed cycle of length $\geq 2k$ if and only if one of the following conditions is satisfied:

- for some n such that $2 \leq n \leq 2k - 3$, G has an elementary directed cycle of length nk; or
- $\widehat{G}$ has a chordless undirected cycle (i.e., a cycle without cycle chord) of length $\geq 2k$.

The first condition can be tested in **FO**; the second condition can be reduced to an undirected connectivity problem, which is in logspace [33] and can be expressed in *SymStratDatalog*. □

The following lemma uses the observation that the strong components of the $\overset{C}{\hookrightarrow}$-graph that contain some n-embedding of C, $n \geq 2$, can be recognized by executing the Datalog program of Lemma 8 on the block-quotient graph.

Lemma 9 *Let q be a query in* sjfBCQ. *Let C be an elementary cycle of length k ($k \geq 2$) in the* M*-graph of q. There exists a program in SymStratDatalog that takes a database* **db** *as input and returns, as output, the maximum garbage set for C in* **db**.

Proof Let the elementary cycle in the M-graph be $C = F_0 \xrightarrow{\mathsf{M}} F_1 \xrightarrow{\mathsf{M}} \cdots \xrightarrow{\mathsf{M}} F_{k-1} \xrightarrow{\mathsf{M}} F_0$ where $k \geq 2$ is the length of the cycle. For each $i \in \{0, \ldots, k-1\}$, let $F_i = R_i(\vec{x}_i, \vec{y}_i)$. Furthermore, for every $i \in \{0, \ldots, k-1\}$ such that the signature of R_i is $[n, \ell]$:

- let $\vec{u}_i$ and $\vec{w}_i$ be sequences of fresh distinct variables of lengths ℓ and $n - \ell$ respectively. Thus, the atom $R_i(\underline{\vec{u}_i}, \vec{w}_i)$ is syntactically well-defined;

– let $\mathsf{RIvantR_i}$ be an IDB predicate of arity n;
– let $\mathsf{GarbageR_i}$ be an IDB predicate of arity ℓ.

Informally, whenever a fact $\mathsf{GarbageR_i}(\vec{a}_i)$ will be derived, then the input database contains a block $R_i(\underline{\vec{a}_i}, *)$ that belongs to the maximum garbage set for C. The stratification of our Datalog program is shown in Fig. 5. We start by defining the IDB predicates $\mathsf{RIvantR_i}$, where $\mathsf{RIvantR_i}(\underline{\vec{a}_i}, \vec{b}_i)$ indicates that $R_i(\underline{\vec{a}_i}, \vec{b}_i)$ belongs to a relevant 1-embedding of C. For every $i \in \{0, 1, \ldots, k-1\}$, we add the rules:

$$\mathsf{RIvantR_i}(\vec{x}_i, \vec{y}_i) \leftarrow q$$

$$\mathsf{GarbageR_i}(\vec{u}_i) \leftarrow R_i(\vec{u}_i, \vec{w}_i), \neg\mathsf{RIvantR_i}(\vec{u}_i, \vec{w}_i)$$

These rules implement Condition 1 in Lemma 5; Condition 5 is also captured since the argument of $\mathsf{GarbageR_i}$ is limited to primary-key positions, which identify blocks rather than individual facts. To implement Condition 4 in Lemma 5, we add, for every $i, j \in \{0, 1, \ldots, k-1\}$ such that $i < j$, the rules:

$$\mathsf{GarbageR_i}(\vec{x}_i) \leftarrow q, \mathsf{GarbageR_j}(\vec{x}_j)$$

$$\mathsf{GarbageR_j}(\vec{x}_j) \leftarrow q, \mathsf{GarbageR_i}(\vec{x}_i)$$

These rules are each other's symmetric version.

For every variable x and every $i \in \{\dagger, \ddagger\} \cup \{0, 1, 2, \ldots\}$, we write $x^{(i)}$ to denote a fresh variable such that $x^{(i)} = y^{(j)}$ if and only if $x = y$ and $i = j$. This notation extends to sequences of variables and queries in the natural way. For example, if $\vec{x} = \langle x_1, x_2, \ldots, x_n \rangle$, then $\vec{x}^{(i)} = \langle {x_1}^{(i)}, {x_2}^{(i)}, \ldots, {x_n}^{(i)} \rangle$. If c is a constant, then we define $c^{(i)} = c$.

We will need to compare composite primary-key values for disequality. To this end, we add the following rules for every $i \in \{0, 1, \ldots, k-1\}$:

$$\mathsf{EqR_i}(\vec{x}_i, \vec{x}_i) \leftarrow R_i(\vec{x}_i, \vec{y}_i)$$

$$\mathsf{NeqR_i}(\vec{x}_i, \vec{x}_i^{(\dagger)}) \leftarrow R_i(\vec{x}_i, \vec{y}_i), R_i(\vec{x}_i^{(\dagger)}, \vec{y}_i^{(\dagger)}), \neg\mathsf{EqR_i}(\vec{x}_i, \vec{x}_i^{(\dagger)})$$

Note that if $R_i(\vec{x}_i, \vec{y}_i)$ contains constants or repeated variables, then it can be falsified by an R_i-fact. The rule for $\mathsf{EqR_i}$, however, only applies to R_i-facts that satisfy the rule body $\{R_i(\vec{x}_i, \vec{y}_i)\}$. This suffices, because R_i-facts falsifying $\{R_i(\vec{x}_i, \vec{y}_i)\}$ cannot belong to a relevant 1-embedding, and will be added to the garbage set by previous rules.

Stratum	Predicates
5	$\mathsf{T}, \mathsf{N}_0, \ldots, \mathsf{N}_{k-1}$
4	IdentifiedBy
3	$\mathsf{KeepR}_0, \ldots, \mathsf{KeepR}_{k-1}$, Link, Trans
2	UCon, InLongUCycle, $\mathsf{GarbageR}_0, \ldots, \mathsf{GarbageR}_{k-1}$
1	Irr1Emb, $\mathsf{NeqR}_0, \ldots, \mathsf{NeqR}_{k-1}$, $\widehat{\mathsf{E}}$
0	Pk, $\mathsf{RIvantR}_0, \ldots, \mathsf{RIvantR}_{k-1}$, $\mathsf{EqR}_0, \ldots, \mathsf{EqR}_{k-1}$, Any1Emb, Rel1Emb, Eq

Fig. 5 Stratification for the symmetric stratified Datalog program constructed in the proofs of Lemmas 9 and 10. The predicate IdentifiedBy is computed by a non-recursive rule that uses aggregation

In what follows, $\mathsf{NeqR_i}(\vec{x}_i, \vec{x}_i^{(\dagger)})$ will be abbreviated as $\vec{x}_i \neq_{R_i} \vec{x}_i^{(\dagger)}$. Likewise, $\mathsf{EqR_i}(\vec{x}_i, \vec{x}_i^{(\dagger)})$ will be abbreviated as $\vec{x}_i =_{R_i} \vec{x}_i^{(\dagger)}$. Of course, in Datalog with $\neq$, these predicates can be expressed by using disequality ($\neq$) instead of negation ($\neg$).

The predicate Any1Emb computes all 1-embeddings of C. Then, Rel1Emb computes the relevant 1-embeddings, and Irr1Emb the irrelevant 1-embeddings, which is needed in the implementation of Condition 2 in Lemma 5. The body of the rule for the IDB predicate Any1Emb is illustrated by Fig. 6. Figure 7 illustrates that k-cycles in the block-quotient graph can be induced by 1-embeddings of C that are not relevant.

$$\mathsf{Any1Emb}(\vec{x}_0^{(0)}, \vec{y}_0^{(0)}, \ldots, \vec{x}_{k-1}^{(k-1)}, \vec{y}_{k-1}^{(k-1)}) \leftarrow \begin{cases} q^{(0)}, q^{(1)}, \ldots, q^{(k-1)}, \\ \vec{x}_0^{(0)} =_{R_0} \vec{x}_0^{(k-1)}, \\ \vec{x}_1^{(1)} =_{R_1} \vec{x}_1^{(0)}, \\ \vec{x}_2^{(2)} =_{R_2} \vec{x}_2^{(1)}, \\ \vdots \\ \vec{x}_{k-1}^{(k-1)} =_{R_{k-1}} \vec{x}_{k-1}^{(k-2)} \end{cases}$$

$$\mathsf{Rel1Emb}(\vec{x}_0, \vec{y}_0, \vec{x}_1, \vec{y}_1, \ldots, \vec{x}_{k-1}, \vec{y}_{k-1}) \leftarrow q$$

$$\mathsf{Irr1Emb}(\vec{x}_0^{(0)}, \ldots, \vec{x}_{k-1}^{(k-1)}) \leftarrow \begin{cases} \mathsf{Any1Emb}(\vec{x}_0^{(0)}, \vec{y}_0^{(0)}, \ldots, \vec{x}_{k-1}^{(k-1)}, \vec{y}_{k-1}^{(k-1)}), \\ \neg\mathsf{Rel1Emb}(\vec{x}_0^{(0)}, \vec{y}_0^{(0)}, \ldots, \vec{x}_{k-1}^{(k-1)}, \vec{y}_{k-1}^{(k-1)}) \end{cases}$$

To finish the implementation of Condition 2 in Lemma 5, we add, for every $i \in \{0, \ldots, k-1\}$, the following rules:

$$\mathsf{GarbageR_i}(\vec{x}_i^{(i)}) \leftarrow \mathsf{Irr1Emb}(\vec{x}_0^{(0)}, \vec{x}_1^{(1)}, \ldots, \vec{x}_{k-1}^{(k-1)})$$

We now add rules that implement the algorithm given in the proof of Lemma 8, capturing Condition 3 in Lemma 5. From here on, whenever C occurs as an

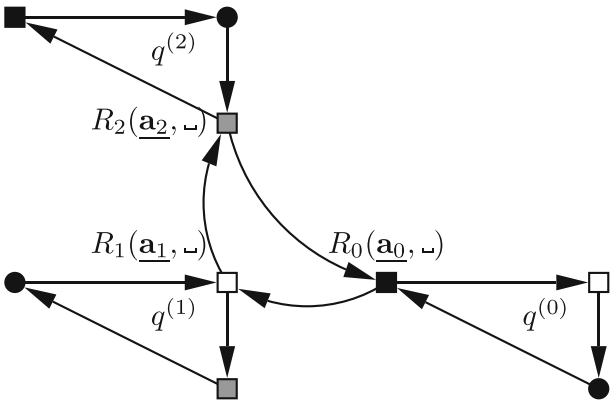
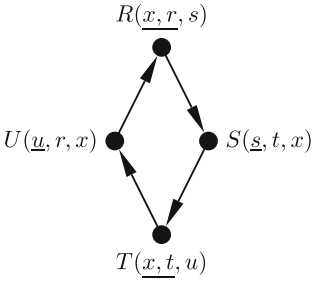
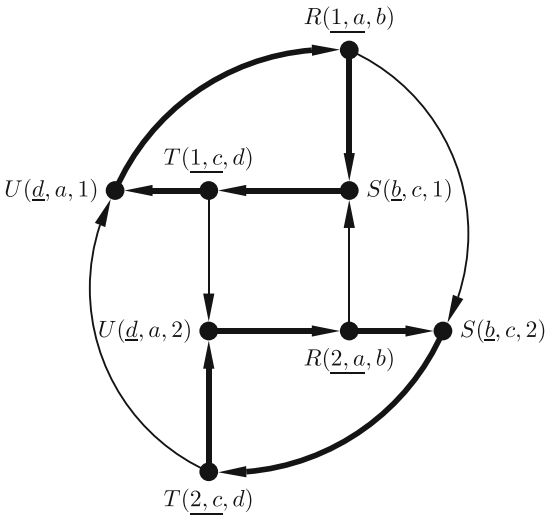


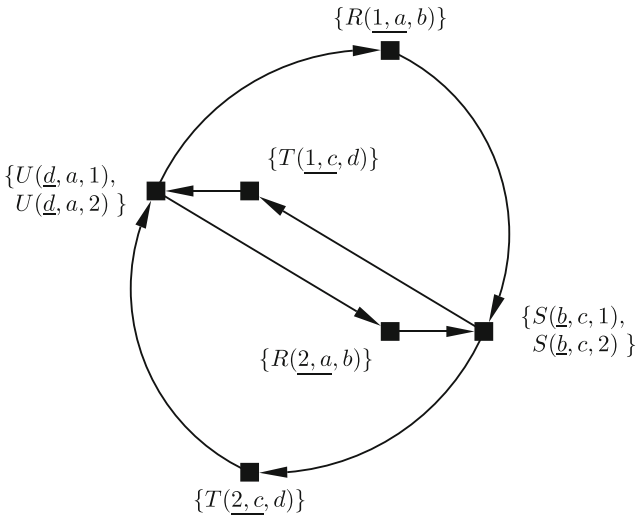
Fig. 6 Illustration of the $\hookrightarrow$-subgraph searched by the body of the rule for Any1Emb with $k = 3$. Rectangular nodes with the same color (white, gray, black) are key-equal. The curved arrows form a k-cycle



(a) M-graph.



(b) $\overset{C}{\hookrightarrow}$-graph.



(c) Block-quotient graph.

Fig. 7 The $\overset{C}{\hookrightarrow}$-graph contains two relevant 1-embeddings (thick arrows) and four irrelevant 1-embeddings. The block-quotient graph contains four elementary cycles of length 4; the outermost cycle (curved arrows) and the innermost cycle (straight arrows) are induced by irrelevant 1-embeddings

argument of a predicate, then it is understood to be a shorthand for the sequence $\langle \vec{x}_0, \vec{x}_1, \ldots, \vec{x}_{k-1} \rangle$. The IDB predicate $\widehat{\mathsf{E}}$ is used for undirected edges between vertices that are k-cycles in the block-quotient graph, and the predicate Eq tests whether two vertices are equal. Figure 8 shows the $\widehat{\mathsf{E}}$-edges for the example of Fig. 2. It suffices to consider only k-cycles of the block-quotient graph induced by relevant 1-embeddings of C, because irrelevant 1-embeddings are already added to the garbage set by previous rules.

$$\mathsf{Eq}(\vec{x}_0, \ldots, \vec{x}_{k-1}; \vec{x}_0, \ldots, \vec{x}_{k-1}) \leftarrow q$$

The use of the semicolon is for readability only. For all $i \in \{0, \ldots, k-1\}$, add the following rules:

$$\widehat{\mathsf{E}}(C^{(0)}, C^{(1)}) \leftarrow \begin{cases} q^{(0)}, q^{(1)}, \neg\mathsf{Eq}(C^{(0)}, C^{(1)}), \\ \vec{x}_i^{(0)} =_{R_i} \vec{x}_i^{(1)} \end{cases}$$

The predicate UCon computes undirected connectivity in the graph defined by $\widehat{\mathsf{E}}$; it takes $2k-1$ vertices as operands, and holds true if there exists an undirected path between the first two operands such that no vertex on the path is equal or adjacent to any of the remaining $2k-3$ operands. In the following rules, we use some self-evident abbreviations; for example,

$$\left\{\neg\mathsf{Eq}(C^{(1)}, C^{(i)}), \neg\widehat{\mathsf{E}}(C^{(1)}, C^{(i)})\right\}_{i=3}^{2k-1}$$

is a shorthand for the following sequence: $\neg\mathsf{Eq}(C^{(1)}, C^{(3)})$, $\neg\widehat{\mathsf{E}}(C^{(1)}, C^{(3)}), \ldots,$ $\neg\mathsf{Eq}(C^{(1)}, C^{(2k-1)})$, $\neg\widehat{\mathsf{E}}(C^{(1)}, C^{(2k-1)})$.

$$\mathsf{UCon}(C^{(1)}, C^{(1)}, C^{(3)}, \ldots, C^{(2k-1)}) \leftarrow \begin{cases} q^{(1)}, q^{(3)}, \ldots, q^{(2k-1)}, \\ \left\{\neg\mathsf{Eq}(C^{(1)}, C^{(i)}), \neg\widehat{\mathsf{E}}(C^{(1)}, C^{(i)})\right\}_{i=3}^{2k-1} \end{cases}$$

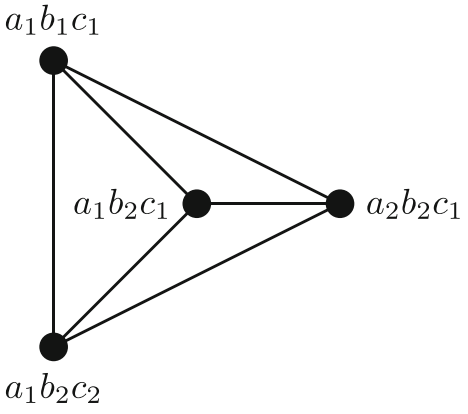


Fig. 8 $\widehat{\mathsf{E}}$-edges for the example of Fig. 2, where $k = 3$. There is no chordless cycle of length $2k = 6$. However, since the inequalities $2 \le n \le 2k-3$ have solutions $n = 2$ and $n = 3$, the Datalog program will also contain non-recursive rules for detecting 2-embeddings and 3-embeddings

$$\mathsf{UCon}(C^{(1)}, C^{(2)}, C^{(3)}, \ldots, C^{(2k-1)}) \leftarrow \begin{cases} \mathsf{UCon}(C^{(1)}, C^{(\dagger)}, C^{(3)}, \ldots, C^{(2k-1)}), \\ \widehat{\mathsf{E}}(C^{(\dagger)}, C^{(2)}), \\ \left\{\neg\mathsf{Eq}(C^{(\dagger)}, C^{(i)}), \neg\widehat{\mathsf{E}}(C^{(\dagger)}, C^{(i)})\right\}_{i=3}^{2k-1}, \\ \left\{\neg\mathsf{Eq}(C^{(2)}, C^{(i)}), \neg\widehat{\mathsf{E}}(C^{(2)}, C^{(i)})\right\}_{i=3}^{2k-1} \end{cases}$$

$$\mathsf{UCon}(C^{(1)}, C^{(\dagger)}, C^{(3)}, \ldots, C^{(2k-1)}) \leftarrow \begin{cases} \mathsf{UCon}(C^{(1)}, C^{(2)}, C^{(3)}, \ldots, C^{(2k-1)}), \\ \widehat{\mathsf{E}}(C^{(\dagger)}, C^{(2)}), \\ \left\{\neg\mathsf{Eq}(C^{(\dagger)}, C^{(i)}), \neg\widehat{\mathsf{E}}(C^{(\dagger)}, C^{(i)})\right\}_{i=3}^{2k-1}, \\ \left\{\neg\mathsf{Eq}(C^{(2)}, C^{(i)}), \neg\widehat{\mathsf{E}}(C^{(2)}, C^{(i)})\right\}_{i=3}^{2k-1} \end{cases}$$

The latter two rules are each other's symmetric version.

We are now ready to encode the two conditions for the existence of an elementary directed cycle of length $\geq 2k$ in the proof of Lemma 8. We add non-recursive rules that detect n-embeddings for every n such that $2 \leq n \leq 2k-3$. We show here only the rules for $n = 2$, i.e., for $\overset{C}{\hookrightarrow}$- cycles of length $2k$ without key-equal atoms. We add, for every $i \in \{0, \ldots, k-1\}$, the following rules, where Pk was defined before by rule (3):

$$\mathsf{GarbageR_i}(\vec{x}_i) \leftarrow \begin{cases} \mathsf{Pk}(\vec{x}_0^{(0)}, \vec{x}_1^{(1)}, \ldots, \vec{x}_{k-1}^{(k-1)}, \vec{x}_0^{(k)}), \\ \mathsf{Pk}(\vec{x}_0^{(k)}, \vec{x}_1^{(k+1)}, \ldots, \vec{x}_{k-1}^{(2k-1)}, \vec{x}_0^{(0)}), \\ \vec{x}_1^{(1)} \neq_{R_1} \vec{x}_1^{(k+1)}, \\ \vec{x}_2^{(2)} \neq_{R_2} \vec{x}_2^{(k+2)}, \\ \vdots \\ \vec{x}_{k-1}^{(k-1)} \neq_{R_{k-1}} \vec{x}_{k-1}^{(2k-1)} \end{cases}$$

The following rule checks whether C belongs to a chordless cycle of length $\geq 2k$.

$$\mathsf{InLongUCycle}(C^{(1)}) \leftarrow \begin{cases} \widehat{\mathsf{E}}(C^{(0)}, C^{(1)}), \widehat{\mathsf{E}}(C^{(1)}, C^{(2)}), \ldots, \widehat{\mathsf{E}}(C^{(2k-1)}, C^{(2k)}), \\ \left\{\neg\widehat{\mathsf{E}}(C^{(i)}, C^{(j)})\right\}_{1 \leq i < i+1 < j \leq 2k-1}, \\ \left\{\neg\mathsf{Eq}(C^{(i)}, C^{(j)})\right\}_{1 \leq i < j \leq 2k-1}, \\ \mathsf{UCon}(C^{(0)}, C^{(2k)}, C^{(2)}, \ldots, C^{(2k-2)}) \end{cases}$$

Finally, to finish the implementation of Condition 3 in Lemma 5, we add, for every $i \in \{0, \ldots, k-1\}$, the following rules:

$$\mathsf{GarbageR_i}(\vec{x}_i) \leftarrow \mathsf{InLongUCycle}(C)$$

This concludes the computation of the maximum garbage set for C. □

Example 5 in Appendix C illustrates the Datalog program of Lemma 9 for the query $q = \{R(\underline{x}, y, z), S(\underline{y}, x, z), U(\underline{z}, a)\}$.

7.4 Elimination of M-Cycles

Given a database **db**, the Datalog program of Lemma 9 allows us to compute the maximum garbage set **o** for C in **db**. The $\overset{C}{\hookrightarrow}$-graph of $\mathbf{db}' := \mathbf{db} \setminus \mathbf{o}$ will be a set of strong components, all initial, each of which is a collection of relevant 1-embeddings of C in $\mathbf{db}'$. The following Lemma 10 introduces a reduction that encodes this $\overset{C}{\hookrightarrow}$-graph by means of a fresh atom $T(\underline{u}, \vec{w})$, where $\mathsf{vars}(\vec{w}) = \mathsf{vars}(C)$ and u is a fresh variable. Whenever $\theta(q) \subseteq \mathbf{db}'$ for some valuation θ over $\mathsf{vars}(q)$, the reduction will add to the database a fact $\mathsf{T}(\underline{cid}, \theta(\vec{w}))$ where *cid* is an identifier for the strong component (in the $\overset{C}{\hookrightarrow}$-graph) that contains $\theta(C)$. Moreover, for every atom $R_i(\underline{\vec{x}_i}, \vec{y}_i)$ in C, the reduction adds $\mathsf{N}_i^{\mathsf{c}}(\underline{\theta(\vec{x}_i)}, cid)$. The $\mathsf{N}_i^{\mathsf{c}}$-relation is consistent, because no R_i-fact belongs to different connected components. The construction is illustrated by Fig. 9. The following lemma captures this reduction and states that it *(i)* is expressible in *SymStratDatalog*$^{\min}$, and *(ii)* does not result in an increase of computational complexity. In the statement of the lemma, if $F_i = R(\underline{\vec{x}_i}, \vec{y}_i)$ for $0 \leq i \leq k-1$, then we can take $T = \mathsf{T}(\underline{u}, \vec{x}_0, \vec{y}_0, \ldots, \vec{x}_{k-1}, \vec{y}_{k-1})$ and $N_i = \mathsf{N}_{\mathsf{i}}^{\mathsf{c}}(\underline{\vec{x}_i}, u)$.

Lemma 10 *Let q be a query in* sjfBCQ. *Let $C = F_0 \xrightarrow{\mathsf{M}} F_1 \xrightarrow{\mathsf{M}} \cdots \xrightarrow{\mathsf{M}} F_{k-1} \xrightarrow{\mathsf{M}} F_0$ with $k \geq 2$ be an elementary cycle in the* M*-graph of q. Let u be a variable such that $u \notin \mathsf{vars}(q)$. Let T be an atom with a fresh relation name such that $\mathsf{key}(T) = \{u\}$ and $\mathsf{vars}(T) = \mathsf{vars}(C) \cup \{u\}$. Let p be a set containing, for every $i \in \{0, \ldots, k-1\}$, an atom N_i of mode* c *with a fresh relation name such that $\mathsf{key}(N_i) = \mathsf{key}(F_i)$ and $\mathsf{vars}(N_i) = \mathsf{key}(F_i) \cup \{u\}$. Then,*

1. *there exists a reduction from the problem* CERTAINTY(q) *to the problem* CERTAINTY$((q \setminus C) \cup \{T\} \cup p)$ *that is expressible in SymStratDatalog*$^{\min}$*; and*
2. *if the attack graph of q contains no strong cycle and some initial strong component of the attack graph contains every atom of $\{F_0, F_1, \ldots, F_{k-1}\}$, then the attack graph of $(q \setminus C) \cup \{T\} \cup p$ contains no strong cycle either.*

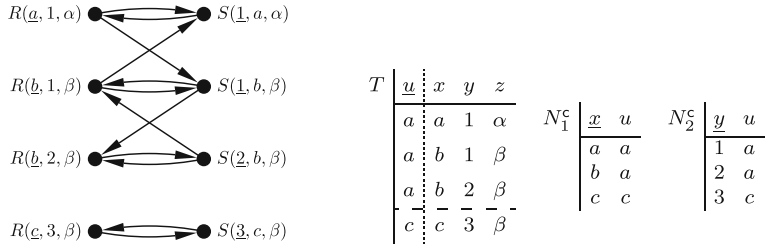


Fig. 9 *Left:* Two strong components in the $\overset{C}{\hookrightarrow}$-graph of a database for an M-cycle $R(\underline{x}, y, z) \xrightarrow{\mathsf{M}} S(\underline{y}, x, z) \xrightarrow{\mathsf{M}} R(\underline{x}, y, z)$. The maximum garbage set is empty. *Right:* Encoding of the relevant 1-embeddings in each strong component. The u-values a and c are used to identify the strong components, and are chosen as the smallest x-values in each strong component

Proof (Crux) The proof of the second item is in Appendix C. The crux in the reduction of the first item is the deterministic choice of u-values for T-blocks. In Fig. 9, for example, the T-block encoding the top strong component uses $u = a$, and the T-block encoding the bottom strong component uses $u = c$. These u-values are the smallest x-values in the strong components, which can be obtained by the query (2) introduced in Section 3. In the example, we assumed $a = \min\{a, b\}$ and $c = \min\{c\}$. We will now show a *SymStratDatalog*$^{\min}$ program for the reduction, the correctness of which is shown in Appendix C.

For every $i \in \{0, \ldots, k-1\}$, let $F_i = R_i(\vec{x}_i, \vec{y}_i)$, and let $\mathsf{KeepR_i}$ be an IDB predicate of the same arity as R_i. For every $i \in \{0, \ldots, k-1\}$, we add the following rule:

$$\mathsf{KeepR_i}(\vec{x}_i, \vec{y}_i) \leftarrow R_i(\vec{x}_i, \vec{y}_i), \neg \mathsf{GarbageR_i}(\vec{x}_i)$$

where $\mathsf{GarbageR_i}$ is the IDB predicate defined in the proof of Lemma 9. Each predicate $\mathsf{KeepR_i}$ is used to compute the R_i-facts that are not in the maximum garbage set.

We now introduce rules for computing the relations for T and for each N_i. For every $i \in \{0, \ldots, k-1\}$, add the rule:

$$\mathsf{Link}(\vec{x}_0, \vec{x}_0^{(\dagger)}) \leftarrow \begin{cases} \mathsf{KeepR_0}(\vec{x}_0, \vec{y}_0), \\ \quad\vdots \\ \mathsf{KeepR_{k-1}}(\vec{x}_{k-1}, \vec{y}_{k-1}), \\ \mathsf{KeepR_0}(\vec{x}_0^{(\dagger)}, \vec{y}_0^{(\dagger)}), \\ \quad\vdots \\ \mathsf{KeepR_{k-1}}(\vec{x}_{k-1}^{(\dagger)}, \vec{y}_{k-1}^{(\dagger)}), \\ \vec{x}_i =_{R_i} \vec{x}_i^{(\dagger)} \end{cases}$$

Informally, a fact $\mathsf{Link}(\vec{a}, \vec{a}')$ tells us that the blocks $R_0(\vec{a}, *)$ and $R_0(\vec{a}', *)$ belong to the same strong component of the $\overset{C}{\hookrightarrow}$-graph. Obviously, Link defines a reflexive and symmetric binary relation on sequences of constants. The predicate Trans computes undirected connectivity in the Link relation.

$$\mathsf{Trans}(\vec{x}_0, \vec{x}_0^{(\dagger)}) \leftarrow \mathsf{Link}(\vec{x}_0, \vec{x}_0^{(\dagger)})$$

$$\mathsf{Trans}(\vec{x}_0, \vec{x}_0^{(\dagger)}) \leftarrow \mathsf{Trans}(\vec{x}_0, \vec{x}_0^{(\ddagger)}), \mathsf{Link}(\vec{x}_0^{(\ddagger)}, \vec{x}_0^{(\dagger)})$$

$$\mathsf{Trans}(\vec{x}_0, \vec{x}_0^{(\ddagger)}) \leftarrow \mathsf{Trans}(\vec{x}_0, \vec{x}_0^{(\dagger)}), \mathsf{Link}(\vec{x}_0^{(\ddagger)}, \vec{x}_0^{(\dagger)})$$

The latter two rules are each other's symmetric version. The following rule picks a single identifier for each connected component of G, using the abbreviated syntax for the query (2) introduced in Section 3.

$$\mathsf{IdentifiedBy}(\vec{x}_0, \min(\vec{x}_0^{(\ddagger)})) \leftarrow \mathsf{Trans}(\vec{x}_0, \vec{x}_0^{(\ddagger)})$$

Informally, $\mathsf{IdentifiedBy}(\vec{a}, \vec{a}')$ means that $\vec{a}'$, rather than $\vec{a}$, will serve to uniquely identify the strong component. The following rule computes all T-facts:

$$\mathsf{T}(\underline{\vec{x}_0^{(\dagger)}}, \vec{x}_0, \vec{y}_0, \ldots, \vec{x}_{k-1}, \vec{y}_{k-1}) \leftarrow \begin{cases} \mathsf{KeepR}_0(\vec{x}_0, \vec{y}_0), \\ \quad\vdots \\ \mathsf{KeepR}_{k-1}(\vec{x}_{k-1}, \vec{y}_{k-1}), \\ \mathsf{IdentifiedBy}(\vec{x}_0, \vec{x}_0^{(\dagger)}) \end{cases}$$

Finally, for every $i \in \{0, \ldots, k-1\}$, add the rule:

$$\mathsf{N}_i(\underline{\vec{x}_i}, \vec{x}_0^{(\dagger)}) \leftarrow \mathsf{T}(\underline{\vec{x}_0^{(\dagger)}}, \vec{x}_0, \vec{y}_0, \ldots, \vec{x}_{k-1}, \vec{y}_{k-1})$$

Note that in this encoding, the cardinality of the primary key of T can be greater than 1. This is not a problem, because we can treat values for u as composite values. □

8 Proof of the Main Theorem

In this section, we will prove Theorem 3, which is the main theoretical contribution of this article. The difficult part of the proof is to show the existence of a consistent *SymStratDatalog*$^{\min}$ rewriting in case of an attack graph without strong cycles and without unattacked atoms. In this case, the proof uses the elimination of M-cycles established in Section 7.4. For this proof to go through, we must first show, in the next subsection, the existence of M-cycles that can be eliminated.

8.1 Saturated Queries

In this section, we distinguish between *saturated* and non-saturated queries in sjfBCQ. It will be shown in Lemma 11 that for every non-saturated query q, there exists a saturated query q' such that *(i)* $\mathsf{CERTAINTY}(q)$ can be first-order reduced to $\mathsf{CERTAINTY}(q')$, and *(ii)* the attack graph of q' contains no strong cycles if the attack graph of q contains no strong cycles. The advantage of saturated queries will become apparent in Lemma 12, which states a crucial property about the existence of M-cycles in saturated queries. This lemma fails for non-saturated queries. The definition of saturated queries refers to a particular type of functional dependencies, which are called *internal*.

Definition 8 Let q be a query in sjfBCQ. Let $Z \to w$ be a functional dependency for q with a singleton right-hand side (where set delimiters { and } are omitted).

A *sequential proof for* $\mathcal{K}(q) \models Z \to w$ is a (possibly empty) sequence $F_1, F_2, \ldots, F_\ell$ of atoms in q such that for every $i \in \{1, \ldots, \ell\}$, $\mathsf{key}(F_i) \subseteq Z \cup \left(\bigcup_{j=1}^{i-1} \mathsf{vars}(F_j)\right)$ and $w \in Z \cup \left(\bigcup_{j=1}^{\ell} \mathsf{vars}(F_j)\right)$. Clearly, if $w \in \mathsf{vars}(F_k)$ for some $k < \ell$, then such a sequential proof can be shortened by omitting the atoms $F_{k+1}, \ldots, F_\ell$. Sequential proofs mimic the computation of a closure of a

set of attributes with respect to a set of functional dependencies; see, for example, [1, p. 165].

We say that $Z \to w$ is *internal to* q if the following two conditions are satisfied:

1. there exists a sequential proof for $\mathcal{K}(q) \models Z \to w$ such that no atom in the sequential proof attacks a variable in $Z \cup \{w\}$; and
2. for some $F \in q$, $Z \subseteq \mathsf{vars}(F)$.

We say that q is *saturated* if for every functional dependency σ that is internal to q, we have $\mathcal{K}(q^{\mathsf{cons}}) \models \sigma$.

Example 4 Consider the query $q = \{S_1(\underline{z}, u),\ S_2(\underline{u}, w),\ R_1(\underline{z}, u'),\ R_2(\underline{u'}, w),\ T_1(\underline{u}, v),\ T_2(\underline{v}, w)\}$. By using relation names as a shorthand for atoms, we have that S_1, S_2 is a sequential proof for $\mathcal{K}(q) \models z \to w$ in which neither S_1 nor S_2 attacks z or w. Indeed, S_1 attacks neither z nor w because $z, w \in {S_1}^{+,q}$. S_2 attacks no variable because $\mathsf{vars}(S_2) \subseteq {S_2}^{+,q}$. It follows that the functional dependency $z \to w$ is internal to q.

Lemma 11 *For every query q in* sjfBCQ*, it is possible to compute a query q' in* sjfBCQ *with the following properties:*

1. *there exists a first-order reduction from* $\mathsf{CERTAINTY}(q)$ *to* $\mathsf{CERTAINTY}(q')$*;*
2. *if the attack graph of q contains no strong cycle, then the attack graph of q' contains no strong cycle; and*
3. *q' is saturated.*

Proof (Sketch) The full proof in Appendix D shows that whenever a functional dependency $\{z_1, \ldots, z_k\} \to w$ is internal to q, then for the query $q' := q \cup \{N^{\mathsf{c}}(\underline{z_1, \ldots, z_k}, w)\}$—where N^{c} is a fresh relation name of mode c—it holds that *(i)* $\mathsf{CERTAINTY}(q)$ can be first-order reduced to $\mathsf{CERTAINTY}(q')$, and *(ii)* the attack graph of q' is free of strong cycles whenever the attack graph of q is free of strong cycles. Informally, this means that every internal functional dependency can be made explicit in a consistent relation. □

The following lemma tells us about how the existence of M-cycles goes hand in hand with weak attack cycles. This relationship is important because, on the one hand, our logspace algorithm for $\mathsf{CERTAINTY}(q)$, with $q \in$ sjfBCQ, is centered on the existence of M-cycles, and, on the other, it must apply whenever all cycles in q's attack graph are weak. The notion of saturated query is also needed here, because the lemma fails for queries that are not saturated. The lemma generalizes Lemma 7.13 in [21]. Note that the lemma only considers strong components of the attack graph that are initial, which will be sufficient for our purposes.

Lemma 12 *Let q be a query in* sjfBCQ *such that q is saturated and the attack graph of q contains no strong cycle. Let $\mathcal{S}$ be an initial strong component in the attack graph of q with $|\mathcal{S}| \geq 2$. Then, the* M*-graph of q contains a cycle all of whose atoms belong to $\mathcal{S}$.*

8.2 Proof of Theorem 3

The proof of the main theorem, Theorem 3, is now fairly straightforward. Informally, let q be a saturated query in sjfBCQ such that the attack graph of q has no strong attack cycles. If q contains an atom of mode i without incoming attacks, then this atom is rewritten in first-order logic, in the form defined by [35, Definition 8.3]; otherwise some M-cycle, which exists by Lemma 12, is eliminated in the way previously described in Section 7.4. In either case, the remaining smaller query will have a consistent *SymStratDatalog*$^{\min}$ rewriting.

Proof of Theorem 3 Let q be a query in sjfBCQ. We can assume that q is saturated; if not, we first apply the reduction of Lemma 11. The first item follows from [21, Theorem 3.2]. In the remainder of the proof, we treat the case that the attack graph of q contains no strong cycle. This case is analogous to the proof of [21, Theorem 7.1], which showed membership in **P**, whereas the current proof shows expressibility in *SymStratDatalog*$^{\min}$. The proof runs by induction on the number of atoms in q that are of mode i. The desired result is obvious if q contains no atom of mode i. Assume next that q contains an atom of mode i. We distinguish two cases.

Case that the Attack graph Contains an Unattacked Atom of Mode i If the attack graph of q contains an unattacked atom of mode i, say $R(\underline{\vec{x}}, \vec{y})$, then it is known (see, for example, [21, Lemma 4.4]) that q is true in every repair only if there exists a valuation θ over $\mathsf{vars}(\vec{x})$ such that $\theta(q)$ is true in every repair. Obviously, if q contains an atom $R(\underline{\vec{a}}, \vec{y})$, where $\vec{a}$ contains no variables, then q is true in every repair only if the input database contains a fact $R(\underline{\vec{a}}, \vec{b})$ such that for every $A \in R(\underline{\vec{a}}, *)$, there exists a valuation θ over $\mathsf{vars}(\vec{y})$ such that $R(\underline{\vec{a}}, \theta(\vec{y})) = A$ and $\theta(q')$ is true in every repair, where $q' = q \setminus \{R(\underline{\vec{x}}, \vec{y})\}$. All this is expressible in first-order logic, and the induction hypothesis applies to $\theta(q')$.

Case that all Atoms of Mode i **are Attacked** Then, every initial strong component of the attack graph contains at least two atoms. By Lemma 12, the M-graph of q has a cycle C all of whose atoms belong to one and the same initial strong component of the attack graph of q. By Lemma 10, there exists a reduction, expressible in *SymStratDatalog*$^{\min}$, from $\mathsf{CERTAINTY}(q)$ to $\mathsf{CERTAINTY}((q \setminus C) \cup \{T\} \cup p)$ such that the attack graph of $(q \setminus C) \cup \{T\} \cup p$ contains no strong cycle. Since the number of atoms of mode i in $(q \setminus C) \cup \{T\} \cup p$ is strictly less than in q (because C is replaced with T and all atoms in p have mode c), by the induction hypothesis, $\mathsf{CERTAINTY}((q \setminus C) \cup \{T\} \cup p)$ is expressible in *SymStratDatalog*$^{\min}$. It follows that $\mathsf{CERTAINTY}(q)$ is expressible in *SymStratDatalog*$^{\min}$.

Thus, in both cases, it is possible to build a consistent *SymStratDatalog*$^{\min}$ rewriting for q by first constructing Datalog rules for an initial strong component of q's attack graph, and then constructing (by induction) a Datalog program for the remainder of the attack graph. The initial strong component has size $= 1$ in the first case, and size > 1 in the second case. All constructed rules together form a consistent *SymStratDatalog*$^{\min}$ rewriting for q. □

9 Joins on Primary Keys

It is common that the join condition in a join of two tables expresses a foreign-to-primary key match, i.e., the columns (called the foreign key) of one table reference the primary key of another table. In our setting, we have primary keys but no foreign keys. Nevertheless, foreign keys can often be inferred from the query. For example, in the following query, the variable d in Movies references the primary key of Directors:

$$\{\mathsf{Movies}(\underline{m}, t, \text{'1963'}, d), \mathsf{Directors}(\underline{d}, \text{'Hitchcock'}, b)\}.$$

Given relation schemas

Movies(M#, Title, Year, Director) and Directors(D#, Name, BirthYear), this query asks whether there exists a movie released in 1963 and directed by Hitchcock.

The *key-join property* that we define below captures this common type of join. Informally, a query has the key-join property if whenever two atoms have a variable in common, then their set of shared variables is either equal to the set of primary-key variables of one of the atoms, or contains all primary-key variables of both atoms.

Definition 9 We say that a query q in sjfBCQ has the *key-join property* if for all $F, G \in q$, either $\mathsf{vars}(F) \cap \mathsf{vars}(G) \in \{\emptyset, \mathsf{key}(F), \mathsf{key}(G)\}$ or $\mathsf{vars}(F) \cap \mathsf{vars}(G) \supseteq \mathsf{key}(F) \cup \mathsf{key}(G)$.

Theorem 4 shows that if a query q in sjfBCQ has the key-join property, then CERTAINTY(q) falls on the logspace side of the dichotomy of Theorem 3.

Theorem 4 *For every query q in* sjfBCQ *that has the key-join property, the problem* CERTAINTY(q) *is expressible in SymStratDatalog*$^{\min}$ *(and therefore is in* **L***).*

It is worth noting that many of the queries covered by Theorem 4 have an acyclic attack graph as well, and therefore even have a consistent first-order rewriting.

10 Conclusion

The main result of this article is that for every query q in sjfBCQ (i.e., the class of self-join-free Boolean conjunctive queries), the problem CERTAINTY(q) is either **coNP**-complete or expressible in *SymStratDatalog*$^{\min}$ (and therefore in **L**). Since CERTAINTY(q) is **L**-complete for some queries q in sjfBCQ, the logspace upper bound in this theorem is tight. The theorem thus culminates a long line of research that started with the ICDT 2005 article of Fuxman and Miller [14]. The outcome of this research is the following theorem.

Theorem 5 *For every self-join-free Boolean conjunctive query q,*

– *if the attack graph of q is acyclic, then* CERTAINTY(q) *is in* **FO***;*

– *if the attack graph of q is cyclic but contains no strong cycle, then the problem* CERTAINTY(q) *is* $\boldsymbol{L}$*-complete and expressible in SymStratDatalog*$^{\min}$*; and*
– *if the attack graph of q contains a strong cycle, then* CERTAINTY(q) *is* **coNP***-complete.*

An intriguing open problem is to extend these complexity results to Boolean conjunctive queries with self-joins and to UCQ. Progress in the latter problem may deepen our understanding of relationships between CQA and CSP, which were first discovered in [13].

Appendix A: Overview of Different Graphs and Notations

Graph	Vertices	Edge Notation	Short Description
attack graph	query atoms	$F \overset{q}{\rightsquigarrow} G$	See Section 3. Informally, $F \overset{q}{\rightsquigarrow} G$ means that there exists a "yes"-instance of CERTAINTY(q) in which two key-equal F-facts join with (and only with) two G-facts that are not key-equal (cf. [35, Proposition 6.4]).
M-graph	query atoms	$F \xrightarrow{\mathsf{M}} G$	Definition 3. Informally, $F \xrightarrow{\mathsf{M}} G$ states that the functional dependency $\mathsf{vars}(F) \rightarrow \mathsf{key}(G)$ is a logical consequence of the primary keys in atoms of mode c.
$\hookrightarrow$-graph	database facts	$A \hookrightarrow B$	Definition 4, data-level instantiation of the M-graph
$\overset{C}{\hookrightarrow}$-graph	database facts	$A \overset{C}{\hookrightarrow} B$	Definition 5, subgraph of the $\hookrightarrow$-graph induced by an M-cycle C
block-quotient graph	database blocks	$(\mathbf{b}, \mathbf{b}')$	Definition 6, quotient graph of the $\overset{C}{\hookrightarrow}$-graph relative to the equivalence relation "is key-equal to"

Notation	Meaning
$\mathsf{key}(F)$	the set of all variables occurring in the primary key of atom F
$\mathsf{vars}(F)$	the set of all variables occurring in atom F
$\mathsf{vars}(q)$	the set of all variables occurring in query q
$\sim$	the equivalence relation "is key-equal to", e.g., $R(\underline{a}, 1) \sim R(\underline{a}, 2)$
$\mathsf{rset}(\mathbf{db})$	the set of all repairs of a database $\mathbf{db}$
$\mathsf{block}(A, \mathbf{db})$	the set of all facts in $\mathbf{db}$ that are key-equal to the fact A
$R(\underline{\vec{a}}, *)$	the set of all database facts of the form $R(\underline{\vec{a}}, \vec{b})$, for some $\vec{b}$
sjfBCQ	the class of self-join-free Boolean conjunctive queries
UCQ	the class of unions of conjunctive queries
R^{c}	a relation name of mode c, which must be interpreted by a consistent relation
q^{cons}	the set of all atoms of query q having a relation name of mode c
$\mathcal{K}(q)$	the set containing $\mathsf{key}(F) \to \mathsf{vars}(F)$ for every $F \in q$
$F^{+,q}$	the closure of $\mathsf{key}(F)$ with respect to the FDs in $\mathcal{K}(q \setminus \{F\}) \cup \mathcal{K}(q^{\mathsf{cons}})$
$\mathsf{genre}_q(A)$	the atom of q with the same relation name as the fact A
$V(G)$	the vertex set of a graph G
$E(G)$	the edge set of a graph G
$\uplus$	a set union that happens to be disjoint

Appendix B: Proofs of Section 5

B.1 Proofs of Lemmas 1 and 2

Proof of Lemma 1 Let $\mathbf{o}_1$ and $\mathbf{o}_2$ be garbage sets for q_0 in $\mathbf{db}$. For every $i \in \{1, 2\}$, we can assume a repair $\mathbf{r}_i$ of $\mathbf{o}_i$ such that

> *Garbage Condition:* for every valuation θ over $\mathsf{vars}(q)$ such that $\theta(q) \subseteq (\mathbf{db} \setminus \mathbf{o}_i) \cup \mathbf{r}_i$, we have $\theta(q_0) \cap \mathbf{r}_i = \emptyset$.

Let $\mathbf{o}_2^- = \mathbf{o}_2 \setminus \mathbf{o}_1$ and $\mathbf{r}_2^- = \mathbf{r}_2 \setminus \mathbf{o}_1$. Then, $\mathbf{r}_1 \uplus \mathbf{r}_2^-$ is a repair of $\mathbf{o}_1 \uplus \mathbf{o}_2^-$, where the use of $\uplus$ (rather than $\cup$) indicates that the operands of the union are disjoint. Let θ be an arbitrary valuation over $\mathsf{vars}(q)$ such that

$$\theta(q) \subseteq \big(\mathbf{db} \setminus (\mathbf{o}_1 \uplus \mathbf{o}_2^-)\big) \cup (\mathbf{r}_1 \uplus \mathbf{r}_2^-).$$

Then, $\theta(q) \subseteq (\mathbf{db} \setminus \mathbf{o}_1) \cup \mathbf{r}_1$. Consequently, by the *Garbage Condition* for $i = 1$, $\theta(q_0) \cap \mathbf{r}_1 = \emptyset$, and therefore $\theta(q_0) \cap \mathbf{o}_1 = \emptyset$. It follows $\theta(q) \subseteq (\mathbf{db} \setminus (\mathbf{o}_1 \cup \mathbf{o}_2)) \cup \mathbf{r}_2^-$, hence $\theta(q) \subseteq (\mathbf{db} \setminus \mathbf{o}_2) \cup \mathbf{r}_2^-$. Consequently, by the *Garbage Condition* for $i = 2$, $\theta(q_0) \cap \mathbf{r}_2^- = \emptyset$. It follows that $\mathbf{o}_1 \uplus \mathbf{o}_2^- = \mathbf{o}_1 \cup \mathbf{o}_2$ is a garbage set for q_0 in $\mathbf{db}$. □

Proof of Lemma 2 The $\Longleftarrow$-direction is trivial. For the $\Longrightarrow$-direction, assume that every repair of $\mathbf{db}$ satisfies q. We can assume a repair $\mathbf{r}_0$ of $\mathbf{o}$ such that for every valuation θ over $\mathsf{vars}(q)$, if $\theta(q) \subseteq (\mathbf{db} \setminus \mathbf{o}) \cup \mathbf{r}_0$, then $\theta(q_0) \cap \mathbf{r}_0 = \emptyset$. Let $\mathbf{r}$ be an arbitrary repair of $\mathbf{db} \setminus \mathbf{o}$. It suffices to show $\mathbf{r} \models q$. Since $\mathbf{r} \cup \mathbf{r}_0$ is a repair of $\mathbf{db}$, we can assume a valuation θ over $\mathsf{vars}(q)$ such that $\theta(q) \subseteq \mathbf{r} \cup \mathbf{r}_0$. Since $\theta(q) \subseteq (\mathbf{db} \setminus \mathbf{o}) \cup \mathbf{r}_0$ is obvious, it follows $\theta(q) \cap \mathbf{r}_0 = \emptyset$. Consequently, $\theta(q) \subseteq \mathbf{r}$, hence $\mathbf{r} \models q$. This concludes the proof. □

B.2 Proof of Lemma 3

We will use two helping lemmas.

Lemma 13 *Let q be a query in* sjfBCQ*, and let $q_0 \subseteq q$. Let $\mathbf{o}$ be a garbage set for q_0 in $\mathbf{db}$. If $\mathbf{p}$ is the union of one or more blocks of $\mathbf{o}$, then $\mathbf{o} \setminus \mathbf{p}$ is a garbage set for q_0 in $\mathbf{db} \setminus \mathbf{p}$.*

Proof Let $\mathbf{p}$ be the union of one or more blocks of $\mathbf{o}$. We can assume a repair $\mathbf{r}$ of $\mathbf{o}$ such that for every valuation θ over $\mathsf{vars}(q)$, if $\theta(q) \subseteq (\mathbf{db} \setminus \mathbf{o}) \cup \mathbf{r}$, then $\theta(q) \cap \mathbf{r} = \emptyset$. Let $\mathbf{s} = \mathbf{r} \setminus \mathbf{p}$. Obviously, $\mathbf{s}$ is a repair of $\mathbf{o} \setminus \mathbf{p}$.

Let θ be a valuation over $\mathsf{vars}(q)$ such that $\theta(q) \subseteq ((\mathbf{db} \setminus \mathbf{p}) \setminus (\mathbf{o} \setminus \mathbf{p})) \cup \mathbf{s}$. It suffices to show $\theta(q) \cap \mathbf{s} = \emptyset$. Since $(\mathbf{db} \setminus \mathbf{p}) \setminus (\mathbf{o} \setminus \mathbf{p}) \subseteq \mathbf{db} \setminus \mathbf{o}$ and $\mathbf{s} \subseteq \mathbf{r}$, it follows $\theta(q) \subseteq (\mathbf{db} \setminus \mathbf{o}) \cup \mathbf{r}$, hence $\theta(q) \cap \mathbf{r} = \emptyset$. It follows $\theta(q) \cap \mathbf{s} = \emptyset$. □

Corollary 1 *Let q be a query in* sjfBCQ*, and let $q_0 \subseteq q$. Let $\mathbf{o}$ be a garbage set for q_0 in $\mathbf{db}$. If every garbage set for q_0 in $\mathbf{db} \setminus \mathbf{o}$ is empty, then $\mathbf{o}$ is the maximum garbage set for q_0 in $\mathbf{db}$.*

Proof Proof by contraposition. Assume that $\mathbf{o}$ is not the maximum garbage set for q_0 in $\mathbf{db}$. Let $\mathbf{o}_0$ be the maximum garbage set for q_0 in $\mathbf{db}$. By Lemma 13, $\mathbf{o}_0 \setminus \mathbf{o}$ is a nonempty garbage set for q_0 in $\mathbf{db} \setminus \mathbf{o}$. □

Lemma 14 *Let q be a query in* sjfBCQ*, and let $q_0 \subseteq q$. Let $\mathbf{db}$ be a database. If $\mathbf{o}$ is a garbage set for q_0 in $\mathbf{db}$, and $\mathbf{p}$ is a garbage set for q_0 in $\mathbf{db} \setminus \mathbf{o}$, then $\mathbf{o} \cup \mathbf{p}$ is a garbage set for q_0 in $\mathbf{db}$.*

Proof Assume the hypothesis holds. Note that $\mathbf{o} \cap \mathbf{p} = \emptyset$. We can assume a repair $\mathbf{r}$ of $\mathbf{o}$ such that for every valuation θ over $\mathsf{vars}(q)$, if $\theta(q) \subseteq (\mathbf{db} \setminus \mathbf{o}) \cup \mathbf{r}$, then $\theta(q) \cap \mathbf{r} = \emptyset$. Likewise, we can assume a repair $\mathbf{s}$ of $\mathbf{p}$ such that for every valuation θ over $\mathsf{vars}(q)$, if $\theta(q) \subseteq ((\mathbf{db} \setminus \mathbf{o}) \setminus \mathbf{p}) \cup \mathbf{s}$, then $\theta(q) \cap \mathbf{s} = \emptyset$. Obviously, $\mathbf{r} \cup \mathbf{s}$ is a repair of $\mathbf{o} \cup \mathbf{p}$.

Let θ be a valuation over $\mathsf{vars}(q)$ such that $\theta(q) \subseteq (\mathbf{db} \setminus (\mathbf{o} \cup \mathbf{p})) \cup (\mathbf{r} \cup \mathbf{s})$. From the set inclusion $(\mathbf{db} \setminus (\mathbf{o} \cup \mathbf{p})) \cup (\mathbf{r} \cup \mathbf{s}) \subseteq (\mathbf{db} \setminus \mathbf{o}) \cup \mathbf{r}$, it follows $\theta(q) \subseteq (\mathbf{db} \setminus \mathbf{o}) \cup \mathbf{r}$, hence $\theta(q) \cap \mathbf{r} = \emptyset$. Then, $\theta(q) \subseteq (\mathbf{db} \setminus (\mathbf{o} \cup \mathbf{p})) \cup \mathbf{s} = ((\mathbf{db} \setminus \mathbf{o}) \setminus \mathbf{p}) \cup \mathbf{s}$, hence $\theta(q) \cap \mathbf{s} = \emptyset$. It follows $\theta(q) \cap (\mathbf{r} \cup \mathbf{s}) = \emptyset$. □

Corollary 2 *Let q be a query in* sjfBCQ*, and let $q_0 \subseteq q$. Let* **db** *be a database, and let* **o** *be the maximum garbage set for q_0 in* **db**. *Then, every garbage set for q_0 in* **db** $\setminus$ **o** *is empty.*

Proof Immediate from Lemma 14. □

The proof of Lemma 3 can now be given.

Proof of Lemma 3 Immediate from Corollaries 1 and 2. □

Appendix C: Appendix to Section 7

C.1 Proofs of Lemmas 5 and 6

Proof of Lemma 5 We will write $\oplus$ for addition modulo k. We first consider garbage sets respecting the first three conditions.

- Let A be a fact of **db** such that $\mathsf{genre}_q(A) \in \{F_0, \ldots, F_{k-1}\}$ and A has zero outdegree in the $\overset{C}{\hookrightarrow}$-graph. Then, there exists no valuation θ over $\mathsf{vars}(q)$ such that $A \in \theta(q) \subseteq$ **db**. It is obvious that $\mathsf{block}(A, \mathbf{db})$ is a garbage set for C in **db**.
- Let $A_0 \overset{C}{\hookrightarrow} A_1 \overset{C}{\hookrightarrow} \cdots \overset{C}{\hookrightarrow} A_{k-1} \overset{C}{\hookrightarrow} A_0$ be an irrelevant 1-embedding of C in **db**. Assume without loss of generality that for every $i \in \{0, \ldots, k-1\}$, $\mathsf{genre}_q(A_i) = F_i$. Let $\mathbf{o} = \bigcup_{i=0}^{k-1} \mathsf{block}(A_i, \mathbf{db})$. Let $\mathbf{r} = \{A_0, \ldots, A_{k-1}\}$, which is obviously a repair of **o**. We show that **o** is a garbage set for C in **db**. Assume, toward a contradiction, the existence of a valuation θ over $\mathsf{vars}(q)$ such that for some $i \in \{0, \ldots, k-1\}$, $A_i \in \theta(q) \subseteq (\mathbf{db} \setminus \mathbf{o}) \cup \mathbf{r}$. Then, $\theta(F_i) \overset{C}{\hookrightarrow} \theta(F_{i\oplus 1})$. Since $\theta(F_i) = A_i$, we have $A_i \overset{C}{\hookrightarrow} \theta(F_{i\oplus 1})$. From $A_i \overset{C}{\hookrightarrow} \theta(F_{i\oplus 1})$ and $A_i \overset{C}{\hookrightarrow} A_{i\oplus 1}$, it follows $\theta(F_{i\oplus 1}) \sim A_{i\oplus 1}$ by Lemma 4. Since $\theta(F_{i\oplus 1}) \in (\mathbf{db} \setminus \mathbf{o}) \cup \mathbf{r}$, it follows $\theta(F_{i\oplus 1}) = A_{i\oplus 1}$. By repeated application of the same reasoning, for every $j \in \{0, \ldots, k-1\}$, $\theta(F_j) = A_j$. But then $A_0 \overset{C}{\hookrightarrow} A_1 \overset{C}{\hookrightarrow} \cdots \overset{C}{\hookrightarrow} A_{k-1} \overset{C}{\hookrightarrow} A_0$ is a relevant 1-embedding of C in **db**, a contradiction.
- Let **r** be a set containing all (and only) the facts of some n-embedding of C in **db** with $n \geq 2$. Let $\mathbf{o} = \bigcup_{A \in \mathbf{r}} \mathsf{block}(A, \mathbf{db})$. It can be shown that **o** is a garbage set for C in **db**; the argumentation is analogous to the reasoning in the previous paragraph.

Let $\mathbf{o}_0$ be the minimal subset of **db** that satisfies all conditions in the statement of the lemma except the recursive Condition 4. By Lemma 1 and our reasoning in the previous items, it follows that $\mathbf{o}_0$ is a garbage set for C in **db**.

Note that the first three conditions do not recursively depend on $\mathbf{o}_0$. Starting with $\mathbf{o}_0$, construct a maximal sequence

$$\mathbf{o}_0, \mu_0, \mathbf{o}_1, \mu_1, \mathbf{o}_2, \mu_2, \ldots, \mathbf{o}_m, \mu_m, \mathbf{o}_{m+1}$$

such that $\mathbf{o}_0 \subsetneq \mathbf{o}_1 \subsetneq \mathbf{o}_2 \subsetneq \cdots \subsetneq \mathbf{o}_{m+1}$ and for every $h \in \{0, 1, \ldots, m\}$,

1. μ_h is a valuation over $\mathsf{vars}(q)$ such that $\mu_h(q) \subseteq \mathbf{db}$ and $\mu_h(q) \cap \mathbf{o}_h \neq \emptyset$. Therefore, $\mu(F_0) \overset{C}{\hookrightarrow} \mu(F_1) \overset{C}{\hookrightarrow} \cdots \overset{C}{\hookrightarrow} \mu(F_{k-1}) \overset{C}{\hookrightarrow} \mu(F_0)$ is a relevant 1-embedding of C in $\mathbf{db}$; and
2. $\mathbf{o}_{h+1} = \mathbf{o}_h \cup \left(\bigcup_{i=0}^{k-1} \mathsf{block}(\mu_h(F_i), \mathbf{db})\right)$.

It is clear that the final set $\mathbf{o}_{m+1}$ is a minimal set satisfying all conditions in the statement of the lemma. We show by induction on increasing h that for all $h \in \{0, 1, \ldots, m, m+1\}$, $\mathbf{o}_h$ is a garbage set for C in $\mathbf{db}$. We have already showed that $\mathbf{o}_0$ is a garbage set for C in $\mathbf{db}$. For the induction step, $h \to h+1$, the induction hypothesis is that $\mathbf{o}_h$ is a garbage set for C in $\mathbf{db}$. Then, there exists a repair $\mathbf{r}$ of $\mathbf{o}_h$ such that for every valuation θ over $\mathsf{vars}(q)$, if $\theta(q) \subseteq (\mathbf{db} \setminus \mathbf{o}_h) \cup \mathbf{r}$, then $\theta(q) \cap \mathbf{r} = \emptyset$.

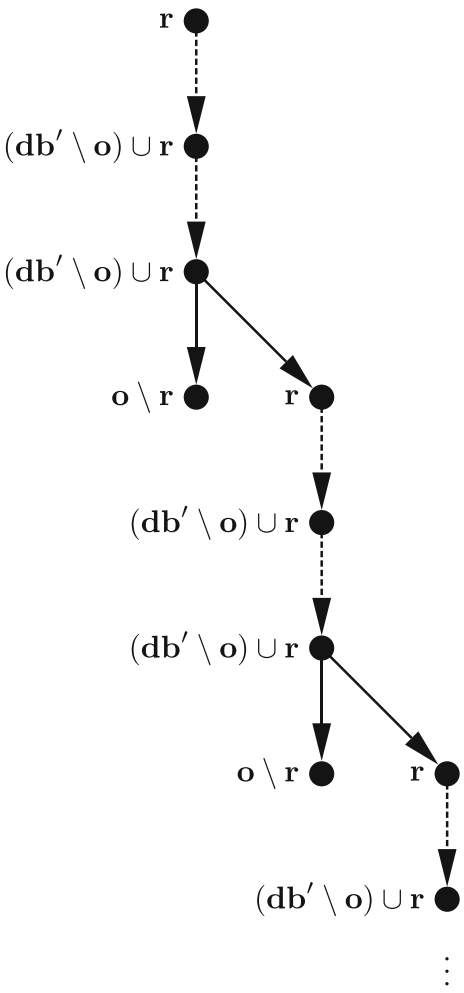


Fig. 10 Illustration of the $\overset{C}{\hookrightarrow}$-graph in the proof of Lemma 5. Every vertex is a fact, and the vertex labels indicate the set to which each vertex belongs. Vertices on the same horizontal line are key-equal. Dashed arrows represent (possibly empty) directed paths

For every $i \in \{0, \ldots, k-1\}$, define $A_i := \mu_h(F_i)$. Let $\mathbf{s} = \{A_0, \ldots, A_{k-1}\} \setminus \mathbf{o}_h$. We have $\mathbf{o}_{h+1} = \mathbf{o}_h \uplus \left(\bigcup_{A_j \in \mathbf{s}} \mathsf{block}(A_j, \mathbf{db})\right)$. Let $\mathbf{r}' = \mathbf{r} \uplus \mathbf{s}$. Obviously, $\mathbf{r}'$ is a repair of $\mathbf{o}_{h+1}$. Here, we use $\uplus$, rather than $\cup$, to make clear that the operands of the union are disjoint. Assume, toward a contradiction, the existence of a valuation θ over $\mathsf{vars}(q)$ such that $\theta(q) \subseteq (\mathbf{db} \setminus \mathbf{o}_{h+1}) \cup \mathbf{r}'$ and $\theta(q) \cap \mathbf{r}' \neq \emptyset$. Since $(\mathbf{db} \setminus \mathbf{o}_{h+1}) \cup \mathbf{r}' \subseteq (\mathbf{db} \setminus \mathbf{o}_h) \cup \mathbf{r}$, it follows $\theta(q) \subseteq (\mathbf{db} \setminus \mathbf{o}_h) \cup \mathbf{r}$, hence $\theta(q) \cap \mathbf{r} = \emptyset$ by our initial hypothesis. It must be the case that $\theta(q) \cap \mathbf{s} \neq \emptyset$. We can assume $i \in \{0, \ldots, k-1\}$ such that $A_i \in \theta(q) \cap \mathbf{s}$. We have $\theta(F_i) \overset{C}{\hookrightarrow} \theta(F_{i\oplus 1})$. Since $\theta(F_i) = A_i$, we have $A_i \overset{C}{\hookrightarrow} \theta(F_{i\oplus 1})$. From $A_i \overset{C}{\hookrightarrow} \theta(F_{i\oplus 1})$ and $A_i \overset{C}{\hookrightarrow} A_{i\oplus 1}$, it follows $\theta(F_{i\oplus 1}) \sim A_{i\oplus 1}$ by Lemma 4. Therefore, $\theta(F_{i\oplus 1}) \in \mathsf{block}(A_{i\oplus 1}, \mathbf{db})$. Two cases are possible:

Case that $\mathsf{block}(A_{i\oplus 1}, \mathbf{db}) \subseteq \mathbf{o}_h$. Since $\theta(F_{i\oplus 1}) \in (\mathbf{db} \setminus \mathbf{o}_h) \cup \mathbf{r}$, it must be the case that $\theta(F_{i\oplus 1}) \in \mathbf{r}$. However, since we have previously argued that $\theta(q) \cap \mathbf{r} = \emptyset$, we conclude that this case cannot occur.

Case that $\mathsf{block}(A_{i\oplus 1}, \mathbf{db}) \not\subseteq \mathbf{o}_h$. By our definition of $\mathbf{s}$, we have $A_{i\oplus 1} \in \mathbf{s}$. Since $\theta(F_{i\oplus 1}) \in (\mathbf{db} \setminus \mathbf{o}_{h+1}) \cup \mathbf{r}'$, it must be the case that $\theta(F_{i\oplus 1}) \in \mathbf{s}$, and therefore $\theta(F_{i\oplus 1}) = A_{i\oplus 1}$.

From the above cases, it follows that $A_{i\oplus 1} \in \theta(q) \cap \mathbf{s}$. By repeating the same reasoning, we obtain that $A_j \in \theta(q) \cap \mathbf{s}$ for all $j \in \{0, \ldots, k-1\}$. Since $\mu_h(q) \cap \mathbf{o}_h \neq \emptyset$ by our construction, we can assume the existence of $\ell \in \{0, \ldots, k-1\}$ such that $A_\ell \in \mathbf{o}_h$, hence $A_\ell \notin \mathbf{s}$, which contradicts our earlier finding that each A_j belongs to $\theta(q) \cap \mathbf{s}$. This concludes the induction step. It is correct to conclude that $\mathbf{o}_{m+1}$ is a garbage set for C in $\mathbf{db}$.

Let $\mathbf{db}' = \mathbf{db} \setminus \mathbf{o}_{m+1}$. We show that the garbage set for C in $\mathbf{db}'$ is empty. Assume, toward a contradiction, that $\mathbf{o}$ is a nonempty garbage set for C in $\mathbf{db}'$. We can assume a repair $\mathbf{r}$ of $\mathbf{o}$ such that for every valuation θ over $\mathsf{vars}(q)$, if $\theta(q) \subseteq (\mathbf{db}' \setminus \mathbf{o}) \cup \mathbf{r}$, then $\theta(q) \cap \mathbf{r} = \emptyset$.

We show that for any $A \in \mathbf{r}$, the $\overset{C}{\hookrightarrow}$-graph contains an infinite path that starts from A such that any vertex on the path belongs to $(\mathbf{db}' \setminus \mathbf{o}) \cup \mathbf{r}$ and any (contiguous) subpath of length k contains some fact from $\mathbf{r}$. To this end, let A be a fact of $\mathbf{r}$. By our construction, there exists a valuation μ over $\mathsf{vars}(q)$ such that $A \in \mu(q) \subseteq \mathbf{db}'$ (otherwise A would belong to $\mathbf{o}_{m+1}$). Hence, $\mu(F_0) \overset{C}{\hookrightarrow} \mu(F_1) \overset{C}{\hookrightarrow} \cdots \overset{C}{\hookrightarrow} \mu(F_{k-1}) \overset{C}{\hookrightarrow} \mu(F_0)$ is a relevant 1-embedding of C in $\mathbf{db}'$ that contains A. Then, for some $i \in \{0, \ldots, k-1\}$, it must be the case that $\mu(F_i) \notin (\mathbf{db}' \setminus \mathbf{o}) \cup \mathbf{r}$ (or else $\mu(q) \subseteq (\mathbf{db}' \setminus \mathbf{o}) \cup \mathbf{r}$ and $\mu(q) \cap \mathbf{r} \neq \emptyset$, a contradiction). Therefore, the $\overset{C}{\hookrightarrow}$-graph contains a shortest path π of length $< k$ from A to some fact $B \in \mathbf{o} \setminus \mathbf{r}$. Then, there exists $B' \in \mathbf{r}$ such that $B' \sim B$ and the $\overset{C}{\hookrightarrow}$-graph contains a path of length $< k$ from A to B'. This path is obtained by substituting B' for B in π. Since $B' \in \mathbf{r}$, we can continue the path by applying the same reasoning as for A. The path is illustrated by Fig. 10. Since the directed path is infinite, it has a shortest finite subpath of length $\geq k$ whose first vertex is key-equal to its last vertex. Let D be the last but one vertex on this subpath. Since the $\overset{C}{\hookrightarrow}$-graph contains a directed edge from D to the first

vertex of the subpath, it contains a cycle of some length nk with $n \geq 1$. Since this cycle is obviously an n-embedding of C in $\mathbf{db}' = \mathbf{db} \setminus \mathbf{o}_{m+1}$, it must be a relevant 1-embedding of C in $\mathbf{db}'$ which, moreover, contains some fact of $\mathbf{r}$. Therefore, there exists a valuation μ over $\mathsf{vars}(q)$ such that $\mu(q) \subseteq (\mathbf{db}' \setminus \mathbf{o}) \cup \mathbf{r}$ and $\mu(q) \cap \mathbf{r} \neq \emptyset$, a contradiction.

Since the garbage set for $\mathbf{db} \setminus \mathbf{o}_{m+1}$ is empty, it follows by Lemma 3 that $\mathbf{o}_{m+1}$ is the maximum garbage set for C in $\mathbf{db}$. This concludes the proof. □

Proof of Lemma 6 For the first item, let $A \overset{C}{\hookrightarrow} A'$ be any edge of the n-embedding. We can assume $F, F' \in C$ such that $F \xrightarrow{\mathsf{M}} F'$, $\mathsf{genre}_q(A) = F$, and $\mathsf{genre}_q(A') = F'$. Then, the block-quotient graph will contain a directed edge from $\mathsf{block}(A, \mathbf{db})$ to $\mathsf{block}(A', \mathbf{db})$. It is then obvious that $(\mathbf{b}_0, \mathbf{b}_1, \ldots, \mathbf{b}_{nk-1}, \mathbf{b}_0)$ is a directed cycle in the block-quotient graph; this cycle is elementary because no two distinct facts of an n-embedding are key-equal.

For the second item, let $i \in \{0, \ldots, nk-1\}$. Since $(\mathbf{b}_i, \mathbf{b}_{i+1 \bmod nk})$ is an edge in the block-quotient graph, we can assume $A_i \in \mathbf{b}_i$ and $A' \in \mathbf{b}_{i+1 \bmod nk}$ such that $A_i \overset{C}{\hookrightarrow} A'$. By Lemma 4, it will be the case that $A_0 \overset{C}{\hookrightarrow} A_1 \overset{C}{\hookrightarrow} \cdots \overset{C}{\hookrightarrow} A_{nk-1} \overset{C}{\hookrightarrow} A_0$. Furthermore, the latter $\overset{C}{\hookrightarrow}$-cycle is an n-embedding. Indeed, since the cycle $(\mathbf{b}_0, \mathbf{b}_1, \ldots, \mathbf{b}_{nk-1}, \mathbf{b}_0)$ is elementary, no two distinct A_is are key-equal. This concludes the proof. □

C.2 Proof of Lemma 8

We will use the following helping lemma. If G is a directed graph, then a directed cycle in G of length k is called a *k-cycle*.

Lemma 15 *Let $G = (V, E)$ be an instance of LONGCYCLE(k). Let $\widehat{G} = (\widehat{V}, \widehat{E})$ be the undirected graph whose vertices are the k-cycles of G. There is an undirected edge between any two distinct k-cycles P_1 and P_2 if $V(P_1) \cap V(P_2) \neq \emptyset$. Then, the following are equivalent:*

1. *$\widehat{G}$ has a chordless cycle of length $\geq 2k$ or G has an elementary directed cycle of length nk with $2 \leq n \leq 2k-3$.*
2. *G contains an elementary directed cycle of length $\geq 2k$.*

Proof Since the graph G is k-partite, every k-cycle is elementary.

$\boxed{1 \Longrightarrow 2}$ Assume that 1 holds true. The result is obvious if there exists n such that $2 \leq n \leq 2k-3$ and G has an elementary cycle of length nk. Assume next that $\widehat{G}$ has a chordless elementary cycle $(P_0, P_1, \ldots, P_{m-1}, P_0)$ of length $m \geq 2k$. We construct a cycle C in G using the following procedure. The construction will define a labeling function ℓ from the vertices in C to $\{0, 1, \ldots, m-1\}$. It will be the case that $w \in V(P_{\ell(w)})$ for every vertex w in C. We start with any vertex $v_0 \in V(P_{m-1}) \cap V(P_0)$ and define its label as $\ell(v_0) := 0$. At any point of the procedure, if we are at vertex u with label $\ell(u)$, we choose the next vertex w in C to be the next vertex in the k-cycle $P_{\ell(u)}$. If $\ell(u) < m-1$ and w also belongs to $P_{\ell(u)+1}$, we let $\ell(w) := \ell(u) + 1$;

otherwise $\ell(w) := \ell(u)$. The procedure terminates when we attempt to add a vertex that already exists in C, and therefore C will be elementary.

We first show that the termination condition will not be met for any vertex distinct from v_0. Suppose, toward a contradiction, that the sequence constructed so far is $C = \langle v_0, v_1, \ldots, v_n \rangle$, $\ell(v_n) = i \leq m-1$, and the next vertex in P_i is some v_j with $j \in \{1, \ldots, n-1\}$. Since v_j belongs to both P_i and $P_{\ell(v_j)}$, it must be the case that $\ell(v_j) \geq i-1$, because otherwise $\{P_i, P_{\ell(v_j)}\}$ is a chord in $(P_0, P_1, \ldots, P_{m-1}, P_0)$, a contradiction. We now distinguish two cases:

Case $\ell(v_j) = i-1$. Then, $v_j \in V(P_{i-1}) \cap V(P_i)$. By the procedure, this means that $\ell(v_{j-1}) = i-2$. Indeed, if $\ell(v_{j-1}) = i-1$, then the procedure would have set $\ell(v_j)$ to i, because v_j also belongs to P_i. But then this also implies that $v_j \in V(P_{i-2})$, a contradiction to the fact that the cycle is chordless.

Case $\ell(v_j) = i$. Then the procedure reaches a vertex on P_i that has been visited before. Therefore, starting with this previously visited vertex on P_i, the procedure has entirely traversed P_i without ever reaching a vertex of $P_{i+1 \mod m}$, contradicting that P_i and $P_{i+1 \mod m}$ have a vertex in common.

It is now clear that at some point we will reach v_0. Indeed, when the label becomes $m-1$, the procedure will follow the edges of P_{m-1} until it reaches v_0. We have that $\ell(v_0) = 0$, and the procedure is such that if some vertex has label i with $i < m-1$, then there is a vertex with label $i+1$. Therefore, for every $i \in \{0, 1, \ldots, m-1\}$, there exists at least one vertex u in C such that $\ell(u) = i$. Therefore, C has at least m vertices. Since $m \geq 2k$, the cycle C has length $\geq 2k$.

$\boxed{2 \Longrightarrow 1}$ Assume that

- G contains an elementary directed cycle of length $\geq 2k$, and
- for all $2 \leq n \leq 2k-3$, G contains no elementary directed cycle of length nk.

We will show that $\widehat{G}$ contains a chordless cycle of length $\geq 2k$.

We first introduce some notions that will be useful in the proof. A subpath of a directed path is a consecutive subsequence of edges of that path. Every path is a subpath of itself. We write $\mathsf{start}(\pi)$ and $\mathsf{end}(\pi)$ to denote, respectively, the first and the last vertex of a directed path π. If $\mathsf{end}(\pi) = \mathsf{start}(\pi')$, then $\pi \cdot \pi'$ denotes the concatenation of paths π and π'. The length of a (possibly closed) elementary path π is the number of edges it contains, and is denoted $\mathit{length}(\pi)$.

Covering Let O be an elementary cycle in G of size $\geq 2k$. A *seam in* O is a subpath of O that is also a subpath of some k-cycle. Obviously, every seam in O has length $< k$. A *covering of* O is a set of edge-disjoint seams in O such that every edge of O is an edge of some seam in the set. Since every edge of G belongs to some k-cycle by our hypothesis, O has a covering. We define $\mathit{seamlength}(O) := \ell$ if O has a covering of cardinality ℓ and every covering of O has cardinality $\geq \ell$.

Cyclic Ordering of the Seams in a Covering Let $C = \{S_0, S_1, \ldots, S_{\ell-1}\}$ be a covering of O. From here on, we will assume that the seams are listed such that a

traversal of O that starts with $\mathsf{start}(S_0)$ traverses the seams of C in the order $S_0, S_1, \ldots, S_{\ell-1}$.

Let O be a directed cycle of length $\geq 2k$ that minimizes *seamlength*$(\cdot)$. From here on, ℓ denotes *seamlength*(O). Thus, every elementary cycle O' in G of length $\geq 2k$ satisfies *seamlength*$(O') \geq \ell$. Let $\{S_0, S_1, \ldots, S_{\ell-1}\}$ be a covering of O.

Our hypothesis is that for every directed cycle of length nk in G such that $n \geq 2$, we have $n > 2k-3$. Consequently, *length*$(O) \geq (2k-2)k$. For every $i \in \{0, \ldots, \ell-1\}$, we have *length*$(S_i) \leq k-1$ (because O is elementary with *length*$(O) \geq 2k$). Therefore, $(2k-2)k \leq$ *length*$(O) = \sum_{i=0}^{\ell-1}$ *length*$(S_i) \leq \ell(k-1)$, which implies $\ell \geq 2k$.

For every $i \in \{0, \ldots, \ell-1\}$, let P_i be a k-cycle of which S_i is a subpath. We define the *fitness of* P_i as *length*(S'_i) if S'_i is the longest subpath of P_i that has S_i as a subpath and that is still a seam in O. Note that the fitness of P_i is at least *length*(S_i). For a reason that will become apparent shortly, if multiple choices for the k-cycle P_i are possible, we will choose a k-cycle with the greatest fitness. Assume, toward a contradiction, that the subgraph of $\widehat{G}$ induced by $\{P_0, P_1, \ldots, P_{\ell-1}\}$ has a cycle chord. We can assume without loss of generality $m \in \{2, \ldots, \ell-2\}$ and a path $(P_0, P_1, \ldots, P_{m-1}, P_m)$ in $\widehat{G}$ such that $\{P_0, P_m\} \in E(\widehat{G})$, while the paths $(P_0, P_1, \ldots, P_{m-1})$ and $(P_1, \ldots, P_{m-1}, P_m)$ are chordless. From $\{P_0, P_m\} \in E(\widehat{G})$, it follows that $V(P_0) \cap V(P_m) \neq \emptyset$. We have $V(S_0) \cap V(S_m) = \emptyset$. Let π be the closed directed path in G that, starting from $\mathsf{start}(S_m)$, traverses P_m until a vertex (call it x) of P_0 is reached. From x on, the path π follows P_0 until $\mathsf{end}(S_0)$ is reached, and then traverses $S_1, S_2, \ldots, S_{m-1}$. Note that it is possible that $x \in V(S_m)$ or $x \in V(S_0)$ (but not both). We argue next that π is an elementary cycle.

The edges of π that are not in O belong either to the subpath (call it π_m) of P_m that goes from $\mathsf{end}(S_m)$ to x, or to the subpath (call it π_0) of P_0 that goes from x to $\mathsf{start}(S_0)$. Note that π_m exists only if $x \notin V(S_m)$, and π_0 exists only if $x \notin V(S_0)$. Assume toward a contradiction that π is not elementary. From our hypotheses and construction, it must be the case that π_m intersects S_{m-1} in some vertex y, or that π_0 intersects S_1 in some vertex z. These possibilities are depicted in Fig 11. If this happens, however, P'_m and P'_0 have a strictly greater fitness than P_m and P_0, contradicting

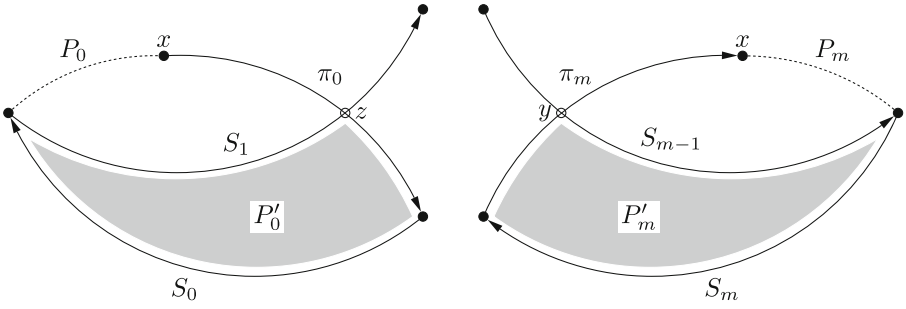


Fig. 11 Two possible configurations. P_0 is the left outermost closed curve containing S_0, x, and z; P_m is the right outermost closed curve containing S_m, y, and x

that we chose k-cycles with the greatest fitness. Here, P'_m is the k-cycle that, starting from $\mathsf{end}(S_{m-1}) = \mathsf{start}(S_m)$, traverses P_m until y, and then follows P_{m-1} from y until $\mathsf{end}(S_{m-1})$. Similarly, P'_0 is the k-cycle that, starting from $\mathsf{end}(S_0) = \mathsf{start}(S_1)$, traverses P_1 until z, and then follows P_0 from z until $\mathsf{end}(S_0)$. To see that P'_m has a strictly greater fitness than P_m, note that the subpath of P'_m from y to $\mathsf{end}(S_m)$ is a seam of O. Since $x \notin V(S_{m-1})$, P_m will cover a strictly smaller suffix of S_{m-1} than P'_m does.

We show that both $length(\pi) = k$ and $length(\pi) \geq 2k$ lead to a contradiction.

- Assume that π is a k-cycle. Then either $S_0 \cdot S_1 \cdot \cdots \cdot S_{m-1}$ is a seam of O or $S_1 \cdot S_2 \cdot \cdots \cdot S_m$ is a seam of O. Since $m \geq 2$, we can use π to construct a covering of O of cardinality $< \ell$, a contradiction.
- Assume that $length(\pi) \geq 2k$. It can be easily seen that π has a covering of cardinality $m + 1 < \ell$, which contradicts our assumption about O. □

The proof of Lemma 8 can now be given.

Proof of Lemma 8 Let $G = (V, E)$ be an instance of LONGCYCLE(k). Let $\widehat{G} = (\widehat{V}, \widehat{E})$ be the undirected graph defined in the statement of Lemma 15. Obviously, it suffices to show that Condition 1 in the statement of Lemma 15 can be expressed in *SymStratDatalog*.

All elementary cycles in G of length nk for $2 \leq n \leq 2k - 3$ can obviously be found in **FO**. We now outline a program in *SymStratDatalog* that tests for the existence of chordless cycles in $\widehat{G}$ of length $\geq 2k$. The graph $\widehat{G}$ can be constructed in *SymStratDatalog*. Then, the existence of a chordless cycle of length $\geq 2k$ can be tested as follows: Check whether there exists a path $(P_0, P_1, P_2, \ldots, P_{2k-2}, P_{2k-1}, P_{2k})$ such that *(i)* the subpath $(P_1, \ldots, P_{2k-1})$ is elementary and chordless, and *(ii)* the endpoints P_0 and P_{2k} are also connected by another (possibly single-vertex) path that uses no vertex that is equal or adjacent to a vertex in $\{P_2, \ldots, P_{2k-2}\}$. In particular, P_0 and P_{2k} themselves must then be distinct from and not adjacent to the vertices in $\{P_2, \ldots, P_{2k-2}\}$, and, consequently, $P_0 \neq P_1$ and $P_{2k} \neq P_{2k-1}$. The single-vertex path occurs if $P_0 = P_{2k}$.

We now give the details of the *SymStratDatalog* program. The following rule states that the vertices of $\widehat{G}$ are the k-cycles of G.

$$\widehat{V}(x_0, \ldots, x_{k-1}) \leftarrow E(x_0, x_1), E(x_1, x_2), \ldots, E(x_{k-2}, x_{k-1}), E(x_{k-1}, x_0)$$

Note incidentally that every k-cycle is stored k times in this way. Since the graph G is k-circle-layered (see Definition 7), we can assume some fixed partition $V_0, V_1, \ldots, V_{k-1}$ of the vertex set V. We will say that the IDB fact $\widehat{V}(a_0, \ldots, a_{k-1})$ is of *class* V_i if $a_0 \in V_i$. Thus, if $\widehat{V}(a_0, a_1, \ldots, a_{k-1})$ is of class V_i, then $\widehat{V}(a_1, \ldots, a_{k-1}, a_0)$ is of class $V_{i+1 \mod k}$. If one partition class would be given as a part of the input, for example as EDB facts $V0(a)$, then an optimization consists in adding $V0(x_0)$ to the body of the previous rule.

We will need an equality test on vertices of $\widehat{G}$:

$$Eq(x_0, \ldots, x_{k-1}; x_0, \ldots, x_{k-1}) \leftarrow \widehat{V}(x_0, \ldots, x_{k-1})$$

The use of the semicolon is for readability only. The following rules compute edges in $\widehat{G}$. For every $\ell \in \{0, \ldots, k-1\}$, add the rules:

$$\widehat{E}(x_0, \ldots, x_{k-1}; y_0, \ldots, y_{k-1}) \leftarrow \begin{cases} \widehat{V}(x_0, \ldots, x_{k-1}), \widehat{V}(y_0, \ldots, y_{k-1}), \\ \neg Eq(x_0, \ldots, x_{k-1}; y_0, \ldots, y_{k-1}), \\ x_\ell = y_\ell \end{cases}$$

Note that whenever $\widehat{E}(a_0, \ldots, a_{k-1}; b_0, \ldots, b_{k-1})$ holds true, then $\widehat{V}(a_0, \ldots, a_{k-1})$ and $\widehat{V}(b_0, \ldots, b_{k-1})$ will be IDB $\widehat{V}$-facts of the same class. In fact, it is sufficient to compute chordless cycles all of whose $\widehat{V}$-facts are of the same class. From here on, we write $\vec{x}$ for the sequence $\langle x_0, \ldots, x_{k-1} \rangle$. Superscripts are used to create new variables: $x^{(i)}$ and $x^{(j)}$ are distinct variables unless $i = j$. Finally, $\vec{x}^{(i)}$ is the sequence ${x_0}^{(i)}, \ldots, {x_{k-1}}^{(i)}$. Likewise for $\vec{y} = \langle y_0, \ldots, y_{k-1} \rangle$, $\vec{z} = \langle z_0, \ldots, z_{k-1} \rangle$, and $\vec{w} = \langle w_0, \ldots, w_{k-1} \rangle$. Add the following rule, as well as its symmetric rule:

$$UCon(\vec{x}, \vec{y}, \vec{z}^{(1)}, \ldots, \vec{z}^{(2k-3)}) \leftarrow \begin{cases} UCon(\vec{x}, \vec{w}, \vec{z}^{(1)}, \ldots, \vec{z}^{(2k-3)}), \widehat{E}(\vec{w}, \vec{y}), \\ \left\{\neg Eq(\vec{w}, \vec{z}^{(i)})\right\}_{i=1}^{2k-3}, \left\{\neg \widehat{E}(\vec{w}, \vec{z}^{(i)})\right\}_{i=1}^{2k-3} \\ \left\{\neg Eq(\vec{y}, \vec{z}^{(i)})\right\}_{i=1}^{2k-3}, \left\{\neg \widehat{E}(\vec{y}, \vec{z}^{(i)})\right\}_{i=1}^{2k-3} \end{cases}$$

$UCon(\vec{a}, \vec{b}, \vec{c}_1, \ldots, \vec{c}_{2k-3})$ holds true if $\widehat{G}$ contains an undirected path between $\vec{a}$ and $\vec{b}$ such that no vertex on the path is equal or adjacent to some $\vec{c}_i$. The basis of the recursion is the following rule:

$$UCon(\vec{x}, \vec{x}, \vec{z}^{(1)}, \ldots, \vec{z}^{(2k-3)}) \leftarrow \begin{cases} \widehat{V}(\vec{x}), \widehat{V}(\vec{z}^{(1)}), \ldots, \widehat{V}(\vec{z}^{(2k-3)}), \\ \left\{\neg Eq(\vec{x}, \vec{z}^{(i)})\right\}_{i=1}^{2k-3}, \left\{\neg \widehat{E}(\vec{x}, \vec{z}^{(i)})\right\}_{i=1}^{2k-3} \end{cases}$$

Finally, the following rule tests for the existence of a chordless cycle in $\widehat{G}$ of length $\geq 2k$.

$$Chordless() \leftarrow \begin{cases} \widehat{E}(\vec{x}^{(0)}, \vec{x}^{(1)}), \widehat{E}(\vec{x}^{(1)}, \vec{x}^{(2)}), \ldots, \widehat{E}(\vec{x}^{(2k-1)}, \vec{x}^{(2k)}), \\ \left\{\neg Eq(\vec{x}^{(i)}, \vec{x}^{(j)})\right\}_{1 \leq i < j \leq 2k-1}, \\ \left\{\neg \widehat{E}(\vec{x}^{(i)}, \vec{x}^{(j)})\right\}_{1 \leq i < i+1 < j \leq 2k-1}, \\ UCon(\vec{x}^{(0)}, \vec{x}^{(2k)}, \vec{x}^{(2)}, \ldots, \vec{x}^{(2k-2)}) \end{cases}$$

This concludes the proof. □

C.3 Illustration of the Datalog Program in the Proof of Lemma 9

The following example illustrates the Datalog program in the proof of Lemma 9.

Example 5 Let $q = \{R(\underline{x}, y, z), S(\underline{y}, x, z), U(\underline{z}, a)\}$, where a is a constant. We show a program in symmetric stratified Datalog that computes the garbage set for the M-cycle $C = R(\underline{x}, y, z) \xrightarrow{\mathsf{M}} S(\underline{y}, x, z) \xrightarrow{\mathsf{M}} R(\underline{x}, y, z)$. In this example, $k = 2$. The program is constructed as in the proof of Lemma 9.

R-facts and S-facts belong to the maximum garbage set if they do not belong to a relevant 1-embedding. This is expressed by the following rules.

$$\mathsf{RlvantR}(x, y, z) \leftarrow R(x, y, z), S(y, x, z), U(z, a)$$

$$\mathsf{GarbageR}(x) \leftarrow R(x, y, z), \neg\mathsf{RlvantR}(x, y, z)$$

$$\mathsf{RlvantS}(y, x, z) \leftarrow R(x, y, z), S(y, x, z), U(z, a)$$

$$\mathsf{GarbageS}(y) \leftarrow S(y, x, z), \neg\mathsf{RlvantS}(y, x, z)$$

If some R-fact or S-fact of a relevant 1-embedding belongs to the maximum garbage set, then every fact of that 1-embedding belongs to the maximum garbage set. This is expressed by the following rules.

$$\mathsf{GarbageR}(x) \leftarrow R(x, y, z), S(y, x, z), U(z, a), \mathsf{GarbageS}(y)$$

$$\mathsf{GarbageS}(y) \leftarrow R(x, y, z), S(y, x, z), U(z, a), \mathsf{GarbageR}(x)$$

Note that the predicates GarbageR and GarbageS refer to blocks: whenever a fact is added to the garbage set, its entire block is added. The following rules compute irrelevant 1-embeddings.

$$\mathsf{Any1Emb}(x, y, z, y', x', z') \leftarrow \begin{cases} R(x, y, z), S(y, x, z), U(z, a), \\ R(x', y', z'), S(y', x', z'), U(z', a), \\ x = x', y = y' \end{cases}$$

$$\mathsf{Rel1Emb}(x, y, z, y, x, z) \leftarrow R(x, y, z), S(y, x, z), U(z, a)$$

$$\mathsf{Irr1Emb}(x, y') \leftarrow \mathsf{Any1Emb}(x, y, z, y', x', z'), \neg\mathsf{Rel1Emb}(x, y, z, y', x', z')$$

The predicate $\widehat{\mathsf{E}}$ is used for edges between vertices; each vertex is a (x, y)-value. The predicate Eq expresses equality of vertices.

$$\mathsf{Eq}(x, y, x, y) \leftarrow R(x, y, z), S(y, x, z), U(z, a)$$

$$\widehat{\mathsf{E}}(x, y, x', y') \leftarrow \begin{cases} R(x, y, z), S(y, x, z), U(z, a), \\ R(x', y', z'), S(y', x', z'), U(z', a), \\ \neg\mathsf{Eq}(x, y, x', y'), x = x' \end{cases}$$

$$\widehat{\mathsf{E}}(x, y, x', y') \leftarrow \begin{cases} R(x, y, z), S(y, x, z), U(z, a), \\ R(x', y', z'), S(y', x', z'), U(z', a), \\ \neg\mathsf{Eq}(x, y, x', y'), y = y' \end{cases}$$

The predicate UCon is used for undirected connectivity of the $\widehat{\mathsf{E}}$-predicate. In particular, it will be the case that $\mathsf{UCon}(a_1, b_1, a_2, b_2, a_3, b_3)$ holds true if there exists a

path between vertices (a_1, b_1) and (a_2, b_2) such that no vertex on the path is equal or adjacent to (a_3, b_3). Recall that each vertex is itself a pair.

$$\mathsf{UCon}(x_1, y_1, x_1, y_1, x_3, y_3) \leftarrow \begin{cases} R(x_1, y_1, z_1), S(y_1, x_1, z_1), U(z_1, a), \\ R(x_3, y_3, z_3), S(y_3, x_3, z_3), U(z_3, a), \\ \neg\mathsf{Eq}(x_1, y_1, x_3, y_3), \neg\widehat{\mathsf{E}}(x_1, y_1, x_3, y_3) \end{cases}$$

$$\mathsf{UCon}(x_1, y_1, x_2, y_2, x_3, y_3) \leftarrow \begin{cases} \mathsf{UCon}(x_1, y_1, x_\dagger, y_\dagger, x_3, y_3), \widehat{\mathsf{E}}(x_\dagger, y_\dagger, x_2, y_2), \\ \neg\mathsf{Eq}(x_\dagger, y_\dagger, x_3, y_3), \neg\widehat{\mathsf{E}}(x_\dagger, y_\dagger, x_3, y_3), \\ \neg\mathsf{Eq}(x_2, y_2, x_3, y_3), \neg\widehat{\mathsf{E}}(x_2, y_2, x_3, y_3) \end{cases}$$

$$\mathsf{UCon}(x_1, y_1, x_\dagger, y_\dagger, x_3, y_3) \leftarrow \begin{cases} \mathsf{UCon}(x_1, y_1, x_2, y_2, x_3, y_3), \widehat{\mathsf{E}}(x_\dagger, y_\dagger, x_2, y_2), \\ \neg\mathsf{Eq}(x_\dagger, y_\dagger, x_3, y_3), \neg\widehat{\mathsf{E}}(x_\dagger, y_\dagger, x_3, y_3), \\ \neg\mathsf{Eq}(x_2, y_2, x_3, y_3), \neg\widehat{\mathsf{E}}(x_2, y_2, x_3, y_3) \end{cases}$$

The latter two rules are each other's symmetric version. The following rule checks whether a vertex (a_1, b_1) belongs to a chordless $\widehat{\mathsf{E}}$-cycle of length $\geq 2k$.

$$\mathsf{InLongUCycle}(x_1, y_1) \leftarrow \begin{cases} \widehat{\mathsf{E}}(x_0, y_0, x_1, y_1), \widehat{\mathsf{E}}(x_1, y_1, x_2, y_2), \\ \widehat{\mathsf{E}}(x_2, y_2, x_3, y_3), \widehat{\mathsf{E}}(x_3, y_3, x_4, y_4), \\ \neg\widehat{\mathsf{E}}(x_1, y_1, x_3, y_3), \\ \neg\mathsf{Eq}(x_1, y_1, x_2, y_2), \neg\mathsf{Eq}(x_1, y_1, x_3, y_3), \neg\mathsf{Eq}(x_2, y_2, x_3, y_3), \\ \mathsf{UCon}(x_0, y_0, x_4, y_4, x_2, y_2) \end{cases}$$

The following rules add to the maximum garbage sets all R-facts and S-facts that belong to an irrelevant 1-embedding or to a strong component of the $\overset{C}{\hookrightarrow}$-graph that contains an elementary $\overset{C}{\hookrightarrow}$-cycle of length $\geq 2k$. Whenever a fact is added, all facts of its block are added.

$$\mathsf{GarbageR}(x) \leftarrow \mathsf{InLongUCycle}(x, y)$$

$$\mathsf{GarbageS}(y) \leftarrow \mathsf{InLongUCycle}(x, y)$$

$$\mathsf{GarbageR}(x) \leftarrow \mathsf{Irr1Emb}(x, y)$$

$$\mathsf{GarbageS}(y) \leftarrow \mathsf{Irr1Emb}(x, y)$$

This terminates the computation of the garbage set. In general, we have to check the existence of elementary $\overset{C}{\hookrightarrow}$-cycles of length nk with $2 \leq n \leq 2k - 3$. However, for $k = 2$, no such n exists.

C.4 Proof of Lemma 10

Proof of Lemma 10 Let $q' = (q \setminus C) \cup \{T\}$. For every $i \in \{0, 1, \ldots, k-1\}$, let $F_i = R_i(\vec{x_i}, \vec{y_i})$. Here is an informal visual representation of the different queries involved:

$$q \begin{cases} & \boxed{C = \{F_0, \ldots, F_{k-1}\}} \\ \boxed{q \setminus C} & \downarrow \end{cases}$$

$$\boxed{\{T\}} \quad \boxed{p = \{N_0, \ldots, N_{k-1}\}}$$

Proof of the First Item We show the existence of a reduction from CERTAINTY(q) to the problem CERTAINTY($q' \cup p$) that is expressible in *SymStratDatalog*$^{\min}$. We first describe the reduction, and then show that it can be expressed in *SymStratDatalog*$^{\min}$.

Let $\mathbf{db}_0$ be a database that is input to CERTAINTY(q). By Lemma 9, we can compute in symmetric stratified Datalog the maximum garbage set $\mathbf{o}$ for C in $\mathbf{db}_0$. Let $\mathbf{db} = \mathbf{db}_0 \setminus \mathbf{o}$. We know, by Lemma 2, that the problem CERTAINTY(q) has the same answer on instances $\mathbf{db}_0$ and $\mathbf{db}$. Moreover, by Lemma 3, every garbage set for C in $\mathbf{db}$ is empty, which implies, by Lemma 5, that *(i)* every n-embedding of C in $\mathbf{db}$ must be a relevant 1-embedding, and *(ii)* every fact A with $\mathsf{genre}_q(A) \in C$ belongs to a 1-embedding. The reduction will now encode all these 1-embeddings as T-facts.

We show that every directed edge of the $\overset{C}{\hookrightarrow}$-graph belongs to a directed cycle. To this end, take any edge $A \overset{C}{\hookrightarrow} B$. Since every garbage set for C in $\mathbf{db}$ is empty, the $\overset{C}{\hookrightarrow}$-graph contains a relevant 1-embedding containing A, and a relevant 1-embedding containing B. Let A' be the fact such that $A' \overset{C}{\hookrightarrow} B$ is a directed edge in the 1-embedding containing B. Let B' be the fact such that $A \overset{C}{\hookrightarrow} B'$ is a directed edge in the 1-embedding containing A. Since $A \overset{C}{\hookrightarrow} B$ and $A \overset{C}{\hookrightarrow} B'$, it follows $B \sim B'$ by Lemma 4. From $A' \overset{C}{\hookrightarrow} B$ and $B \sim B'$, it follows $A' \overset{C}{\hookrightarrow} B'$. Thus, the $\overset{C}{\hookrightarrow}$-graph contains a directed path from B to A', an edge from A' to B', and a directed path from B' to A. Consequently, the $\overset{C}{\hookrightarrow}$-graph contains a directed path from B to A.

It follows that every strong component of the $\overset{C}{\hookrightarrow}$-graph is initial. It can be easily seen that if an initial strong component contains some fact A, then it contains every fact that is key-equal to A. Let $\mathbf{r}$ be a repair of $\mathbf{db}$. For every fact $A \in \mathbf{r}$, there exists a unique fact $B \in \mathbf{r}$ such that $A \overset{C}{\hookrightarrow} B$. It follows that $\mathbf{r}$ must contain an elementary $\overset{C}{\hookrightarrow}$-cycle, which must be a relevant 1-embedding (because every garbage set for C in $\mathbf{db}$ is empty) belonging to the same initial strong component as A. It can also be seen that there exists a repair that contains exactly one such 1-embedding for every strong component of the $\overset{C}{\hookrightarrow}$-graph.

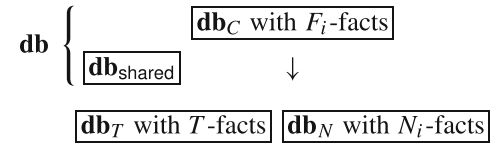
We define an undirected graph G as follows: for each valuation μ over $\mathsf{vars}(q)$ such that $\mu(q) \subseteq \mathbf{db}$, we introduce a vertex θ with $\theta = \mu[\mathsf{vars}(C)]$. We add an edge between two vertices θ and θ' if for some $i \in \{0, \ldots, k-1\}$, $\theta(\vec{x}_i) = \theta'(\vec{x}_i)$. The graph G can clearly be constructed in logarithmic space (and even in **FO**). We define a set $\mathbf{db}_T$ of T-facts and, for every $i \in \{0, \ldots, k-1\}$, a set $\mathbf{db}_i$ as follows: for all two vertices θ, θ' of G, if

$$\theta'(\vec{x}_0) = \min\left\{\theta''(\vec{x}_0) \mid \theta'' \in V(G) \text{ belongs to the same strong component as } \theta\right\},$$

then we add to $\mathbf{db}_T$ the fact $\theta_{[u \mapsto \theta'(\vec{x}_0)]}(T)$, and we add to $\mathbf{db}_i$ the fact $\theta_{[u \mapsto \theta'(\vec{x}_0)]}(N_i)$. In this way, every $\mathbf{db}_i$ is consistent. Informally, if T is the atom $T(\underline{u}, \vec{w})$, then we add to $\mathbf{db}_T$ the T-fact $T(\underline{\theta'(\vec{x}_0)}, \theta(\vec{w}))$, where $\theta'(\vec{x}_0)$ is treated as a single value. This fact represents that θ belongs to the strong component that is identified by $\theta'(\vec{x}_0)$. Since undirected connectivity can be computed in logarithmic space [33], $\mathbf{db}_T$ and each $\mathbf{db}_i$ can be constructed in logarithmic space.

Let $\mathbf{db}_C$ be the set of all F_i-facts in $\mathbf{db}$ ($0 \leq i \leq k-1$), and let $\mathbf{db}_{\mathsf{shared}} := \mathbf{db} \setminus \mathbf{db}_C$, the part of the database $\mathbf{db}$ that is preserved by the reduction. Let $\mathbf{db}_N = \bigcup_{i=0}^{k-1} \mathbf{db}_i$. Since $\mathbf{db}_N$ is consistent, $\mathbf{db}_{\mathsf{shared}} \uplus \mathbf{db}_T \uplus \mathbf{db}_N$ is a legal input to

CERTAINTY($q' \cup p$), where the use of $\uplus$ (rather than $\cup$) indicates that the operands of the union are disjoint. Here is an informal visual representation of the reduction:



We show that the following are equivalent:

1. Every repair of $\mathbf{db}$ satisfies q.
2. For every $\mathbf{s} \in \mathsf{rset}(\mathbf{db}_{\mathsf{shared}})$, for every repair $\mathbf{r}_T$ of $\mathbf{db}_T$, $\mathbf{s} \uplus \mathbf{r}_T \uplus \mathbf{db}_N \models q' \cup p$.
3. Every repair of $\mathbf{db}_{\mathsf{shared}} \uplus \mathbf{db}_T \uplus \mathbf{db}_N$ satisfies $q' \cup p$.

The equivalence $2 \iff 3$ is straightforward. We show next the equivalence $1 \iff 2$. $\boxed{1 \Longrightarrow 2}$ Let $\mathbf{s} \in \mathsf{rset}(\mathbf{db}_{\mathsf{shared}})$ and let $\mathbf{r}_T$ be a repair of $\mathbf{db}_T$. By our construction of $\mathbf{db}_T$, there exists a repair $\mathbf{r}_C$ of $\mathbf{db}_C$ such that for every valuation θ over $\mathsf{vars}(q)$, if $\theta(q) \subseteq \mathbf{s} \cup \mathbf{r}_C$, then for some value c, $\theta_{[u \mapsto c]}(q' \cup p) \subseteq \mathbf{s} \cup \mathbf{r}_T \cup \mathbf{db}_N$. Informally, $\mathbf{r}_C$ contains all (and only) the relevant 1-embeddings of C in $\cup\mathbf{r}_C$ that are encoded by the T-facts of $\mathbf{r}_T$. Since $\mathbf{s} \cup \mathbf{r}_C$ is a repair of $\mathbf{db}$, by the hypothesis 1, we can assume a valuation θ over $\mathsf{vars}(C)$ such that $\theta(q) \subseteq \mathbf{s} \cup \mathbf{r}_C$. Consequently, for some value c, $\theta_{[u \mapsto c]}(q' \cup p) \subseteq \mathbf{s} \cup \mathbf{r}_T \cup \mathbf{db}_N$. $\boxed{2 \Longrightarrow 1}$ Let $\mathbf{r}$ be a repair of $\mathbf{db}$. There exist $\mathbf{s} \in \mathsf{rset}(\mathbf{db}_{\mathsf{shared}})$ and $\mathbf{r}_C \in \mathsf{rset}(\mathbf{db}_C)$ such that $\mathbf{r} = \mathbf{s} \cup \mathbf{r}_C$. By the construction of $\mathbf{db}_T$, there exists a repair $\mathbf{r}_T$ of $\mathbf{db}_T$ such that for every valuation θ over $\mathsf{vars}(q)$, if $\theta_{[u \mapsto c]}(q' \cup p) \subseteq \mathbf{s} \cup \mathbf{r}_T \cup \mathbf{db}_N$ for some c, then $\theta(q) \subseteq \mathbf{s} \cup \mathbf{r}_C$ (note incidentally that the converse does not generally hold). Informally, for every strong component $\mathcal{S}$ of the $\overset{C}{\hookrightarrow}$-graph of $\mathbf{db}$ such that $\mathbf{s} \cup (\mathbf{r}_C \cap V(\mathcal{S})) \models q$, the set $\mathbf{r}_T$ encodes one 1-embedding of C in $\mathbf{s} \cup (\mathbf{r}_C \cap V(\mathcal{S}))$. Here, $V(\mathcal{S})$ denotes the vertex set of the strong component $\mathcal{S}$; thus $V(\mathcal{S}) \subseteq \mathbf{db}_C$. Since $\mathbf{s} \cup \mathbf{r}_T \cup \mathbf{db}_N$ is a repair of $\mathbf{db}_{\mathsf{shared}} \uplus \mathbf{db}_T \uplus \mathbf{db}_N$, it follows by the hypothesis 2 that there exists a valuation θ over $\mathsf{vars}(q)$ such that $\theta_{[u \mapsto c]}(q' \cup p) \subseteq \mathbf{s} \cup \mathbf{r}_T \cup \mathbf{db}_N$ for some c. Consequently, $\theta(q) \subseteq \mathbf{s} \cup \mathbf{r}_C$.

In the main body of this article, we have shown a program in *SymStratDatalog*$^{\min}$ that computes the reduction.

Proof of the Second Item Assume that the attack graph of q contains no strong cycle and that some initial strong component of the attack graph contains every atom of $\{F_0, F_1, \ldots, F_{k-1}\}$. Since all N_i-facts have mode c, they have no outgoing attacks in the attack graph of $q' \cup p$. Since $\mathsf{vars}(N_i) \subseteq \mathsf{vars}(T)$ for every atom $N_i \in p$, we can limit our analysis to witnesses for attacks that do not contain any N_i. Indeed, if N_i would occur in a witness, it can be replaced with T. Let $\mathcal{S}$ be an initial strong component of the attack graph of q that contains every atom of $\{F_0, F_1, \ldots, F_{k-1}\}$. We will use the following properties:

(a) For all $X, Y \subseteq \mathsf{vars}(q)$, if $\mathcal{K}(q) \models X \to Y$, then $\mathcal{K}(q' \cup p) \models X \to Y$. This holds true because $\mathcal{K}(q' \cup p) \models \mathcal{K}(q)$. To prove the latter claim, note that $\mathcal{K}(q) \setminus \mathcal{K}(q' \cup p) = \{\mathsf{key}(F_i) \to \mathsf{vars}(F_i)\}_{i=0}^{k-1}$. For all $i \in \{0, 1, \ldots, k-1\}$,

we have that $\mathcal{K}(\{T, N_i\}) \equiv \{u \rightarrow \mathsf{vars}(C), \mathsf{key}(F_i) \rightarrow u\}$ with $\mathsf{vars}(F_i) \subseteq \mathsf{vars}(C)$. Consequently, $\mathcal{K}(q' \cup p) \models \mathsf{key}(F_i) \rightarrow \mathsf{vars}(F_i)$.

(b) As an immediate consequence of (a), we have $H^{+,q} \subseteq H^{+,q'\cup p}$ for every $H \in q \setminus C$.

(c) For every $H \in q \setminus C$, if $H \overset{q'\cup p}{\rightsquigarrow} T$, then $H \in \mathcal{S}$. To show this result, let $H \in q \setminus C$ such that $H \overset{q'\cup p}{\rightsquigarrow} T$. We can assume without loss of generality the existence of a witness for $H \overset{q'\cup p}{\rightsquigarrow} T$ of the form $\omega \overset{v}{\frown} T$ with $v \neq u$, where the sequence ω starts with H. We can assume the existence of $j \in \{0, \ldots, k-1\}$ such that $v \in \mathsf{vars}(F_j)$. From the preceding property (b), it follows that the sequence $\omega \overset{v}{\frown} F_j$ is a witness for $H \overset{q}{\rightsquigarrow} F_j$. Since $F_j \in \mathcal{S}$, we conclude $H \in \mathcal{S}$.

(d) For all $G, H \in \mathcal{S}$, we have $\mathcal{K}(q' \cup p) \models \mathsf{key}(G) \rightarrow \mathsf{key}(H)$. To show this result, let $G, H \in \mathcal{S}$. Since $\mathcal{S}$ is an initial strong component of the attack graph of q, there exists an elementary attack cycle that contains both G and H. Since the attack graph of q contains no strong cycle, for every edge $J \overset{q}{\rightsquigarrow} J'$ on this attack cycle, we have $\mathcal{K}(q) \models \mathsf{key}(J) \rightarrow \mathsf{key}(J')$. It can now be easily seen that $\mathcal{K}(q) \models \mathsf{key}(G) \rightarrow \mathsf{key}(H)$. Finally, by property (a), $\mathcal{K}(q' \cup p) \models \mathsf{key}(G) \rightarrow \mathsf{key}(H)$.

We know by [21, Lemma 3.6] that if the attack graph contains a strong cycle, then it contains a strong cycle of length 2. Therefore, to conclude the proof, it suffices to show that every cycle of length 2 in the attack graph of $q' \cup p$ is weak. To this end, assume that the attack graph of $q' \cup p$ contains an attack cycle $H \overset{q'\cup p}{\rightsquigarrow} J \overset{q'\cup p}{\rightsquigarrow} H$. Then, either $H \neq T$ or $J \neq T$ (or both). We assume without loss of generality that $H \neq T$. We show that the attack cycle $H \overset{q'\cup p}{\rightsquigarrow} J \overset{q'\cup p}{\rightsquigarrow} H$ is weak. We distinguish three cases.

Case that $H \overset{q'\cup p}{\not\rightsquigarrow} T$ (therefore $J \neq T$) and $J \overset{q'\cup p}{\not\rightsquigarrow} T$. Then no witness for $H \overset{q'\cup p}{\rightsquigarrow} J$ or $J \overset{q'\cup p}{\rightsquigarrow} H$ can contain T. By property (b), $H \overset{q}{\rightsquigarrow} J \overset{q}{\rightsquigarrow} H$. Since the attack graph of q contains no strong attack cycle, $\mathcal{K}(q) \models \mathsf{key}(H) \rightarrow \mathsf{key}(J)$ and $\mathcal{K}(q) \models \mathsf{key}(J) \rightarrow \mathsf{key}(H)$. Then, by property (a), $\mathcal{K}(q' \cup p) \models \mathsf{key}(H) \rightarrow \mathsf{key}(J)$ and $\mathcal{K}(q' \cup p) \models \mathsf{key}(J) \rightarrow \mathsf{key}(H)$. It follows that the attack cycle $H \overset{q'\cup p}{\rightsquigarrow} J \overset{q'\cup p}{\rightsquigarrow} H$ is weak.

Case that $H \overset{q'\cup p}{\rightsquigarrow} T$. By property (c), $H \in \mathcal{S}$. We distinguish two cases.

Case that $J = T$. By property (d), $\mathcal{K}(q' \cup p) \models \mathsf{key}(H) \rightarrow \mathsf{key}(F_0)$ and $\mathcal{K}(q' \cup p) \models \mathsf{key}(F_0) \rightarrow \mathsf{key}(H)$. In the following, recall that $\{u\} = \mathsf{key}(T)$. Since $\mathcal{K}(q' \cup p) \models \mathsf{key}(F_0) \rightarrow u$ and $\mathcal{K}(q' \cup p) \models u \rightarrow \mathsf{key}(F_0)$ hold by the construction of $q' \cup p$, we conclude $\mathcal{K}(q' \cup p) \models \mathsf{key}(H) \rightarrow u$ and $\mathcal{K}(q' \cup p) \models u \rightarrow \mathsf{key}(H)$. It follows that the attack cycle $H \overset{q'\cup p}{\rightsquigarrow} J \overset{q'\cup p}{\rightsquigarrow} H$ is weak.

Case that $J \neq T$. We show that $J \in \mathcal{S}$ by distinguishing two cases:

– If $J \overset{q'\cup p}{\not\rightsquigarrow} T$, then no witness for $J \overset{q'\cup p}{\rightsquigarrow} H$ contains T. Then, by property (b), any witness for $J \overset{q'\cup p}{\rightsquigarrow} H$ is also a witness for $J \overset{q}{\rightsquigarrow} H$, and therefore $J \in \mathcal{S}$.

– If $J \overset{q' \cup p}{\rightsquigarrow} T$, then $J \in \mathcal{S}$ by property (c).

From $H, J \in \mathcal{S}$, it follows $\mathcal{K}(q' \cup p) \models \mathsf{key}(H) \rightarrow \mathsf{key}(J)$ and $\mathcal{K}(q' \cup p) \models \mathsf{key}(J) \rightarrow \mathsf{key}(H)$ by property (d). It follows that the attack cycle $H \overset{q' \cup p}{\rightsquigarrow} J \overset{q' \cup p}{\rightsquigarrow} H$ is weak.

Case that $J \overset{q' \cup p}{\rightsquigarrow} T$ (therefore $J \neq T$). This case is symmetrical to a case that has already been treated. □

Appendix D: Proofs of Section 8.1

D.1 Proof of Lemma 11

We will use two helping lemmas.

Lemma 16 [35, Lemma 4.3] *Let q be a self-join-free Boolean conjunctive query, and $\mathbf{r}$ a consistent database. If α_1, α_2 are valuations over $\mathsf{vars}(q)$ such that $\alpha_1(q) \subseteq \mathbf{r}$ and $\alpha_2(q) \subseteq \mathbf{r}$, then $\{\alpha_1, \alpha_2\}$ satisfies every functional dependency in $\mathcal{K}(q)$.*

Lemma 17 *Let q be a query in* sjfBCQ. *Let $Z \rightarrow w$ be a functional dependency that is internal to q. Let $\vec{z}$ be a sequence of distinct variables such that $\mathsf{vars}(\vec{z}) = Z$. Let $q' = q \cup \{N^{\mathsf{c}}(\underline{\vec{z}}, w)\}$ where N is a fresh relation name of mode* c. *Then,*

1. *there exists a first-order reduction from* CERTAINTY(q) *to* CERTAINTY(q')*; and*
2. *if the attack graph of q contains no strong cycle, then the attack graph of q' contains no strong cycle.*

Proof of the first item By the second condition in Definition 8, we can assume an atom $F \in q$ such that $Z \subseteq \mathsf{vars}(F)$. Let $F_1, F_2, \ldots, F_\ell$ be a sequential proof for $\mathcal{K}(q) \models Z \rightarrow w$ such that for every $i \in \{1, \ldots, \ell\}$, for every $u \in Z \cup \{w\}$, $F_i \overset{q}{\not\rightsquigarrow} u$. It can be easily seen that for every $i \in \{0, \ldots, \ell - 1\}$, we have

$$\mathcal{K}(\{F_j\}_{j=1}^{i}) \models Z \rightarrow \mathsf{key}(F_{i+1}). \tag{4}$$

Let $\mathbf{db}$ be a database that is the input to CERTAINTY(q). We repeat the following "purification" step: If for two valuations over $\mathsf{vars}(q)$, denoted β_1 and β_2, we have $\beta_1(q), \beta_2(q) \subseteq \mathbf{db}$ and $\{\beta_1, \beta_2\} \not\models Z \rightarrow w$, then we remove both the F-block containing $\beta_1(F)$ and the F-block containing $\beta_2(F)$. Note that $\beta_1(F)$ and $\beta_2(F)$ may be key-equal, and hence belong to the same F-block.

Assume that we apply this step on $\mathbf{db}'$ and obtain $\mathbf{db}''$. We show that some repair of $\mathbf{db}'$ falsifies q if and only if some repair of $\mathbf{db}''$ falsifies q. The $\Longrightarrow$-direction trivially holds true. For the $\Longleftarrow$-direction, let $\mathbf{r}''$ be a repair of $\mathbf{db}''$ that falsifies q. Assume, toward a contradiction, that every repair of $\mathbf{db}'$ satisfies q. For every repair

$\mathbf{r}$, define $\mathsf{Reify}(\mathbf{r})$ as the set of valuations over $Z \cup \{w\}$ containing θ if $\mathbf{r} \models \theta(q)$. Let

$$\mathbf{r}' = \begin{cases} \mathbf{r}'' \cup \{\beta_j(F)\} \text{ for some } j \in \{1,2\} & \text{if } \beta_1(F) \text{ and } \beta_2(F) \text{ are key-equal} \\ \mathbf{r}'' \cup \{\beta_1(F), \beta_2(F)\} & \text{otherwise} \end{cases}$$

Note that if $\beta_1(F)$ and $\beta_2(F)$ are key-equal, then we can choose either $\mathbf{r}' = \mathbf{r}'' \cup \{\beta_1(F)\}$ or $\mathbf{r}' = \mathbf{r}'' \cup \{\beta_2(F)\}$; the actual choice does not matter. Obviously, $\mathbf{r}'$ is a repair of $\mathbf{db}'$. Since we assumed that every repair of $\mathbf{db}'$ satisfies q, we can assume a valuation α over $\mathsf{vars}(q)$ such that $\alpha(q) \subseteq \mathbf{r}'$. Since $\alpha(q) \nsubseteq \mathbf{r}''$ (because $\mathbf{r}'' \not\models q$), it must be the case that for some $j \in \{1,2\}$, $\alpha(F) = \beta_j(F)$. From $\mathsf{vars}(\vec{z}) = Z \subseteq \mathsf{vars}(F)$, it follows that $\alpha(\vec{z}) = \beta_j(\vec{z})$. From $\beta_1(\vec{z}) = \beta_2(\vec{z})$, it follows $\alpha(\vec{z}) = \beta_1(\vec{z})$ and $\alpha(\vec{z}) = \beta_2(\vec{z})$. Since $\beta_1(w) \neq \beta_2(w)$, either $\alpha(w) \neq \beta_1(w)$ or $\alpha(w) \neq \beta_2(w)$ (or both). Therefore, we can assume $b \in \{1,2\}$ such that $\alpha(w) \neq \beta_b(w)$. It will be the case that $\mathsf{Reify}(\mathbf{r}') = \{\alpha[Z \cup \{w\}]\}$.[2] Indeed, since α is an arbitrary valuation over $\mathsf{vars}(q)$ such that $\alpha(q) \subseteq \mathbf{r}'$, it follows that for all valuations α_1, α_2 over $\mathsf{vars}(q)$, if $\alpha_1(q), \alpha_2(q) \subseteq \mathbf{r}'$, then $\alpha_1(\vec{z}) = \alpha_2(\vec{z})$ and therefore, by Lemma 16 and using that $\mathcal{K}(q) \models Z \to w$, we have $\alpha_1(w) = \alpha_2(w)$.

We now claim that for all $i \in \{0, 1, \ldots, \ell\}$, there exists a pair $(\mathbf{r}'^i, \alpha^i)$ such that

1. $\mathbf{r}'^i$ is a repair of $\mathbf{db}'$;
2. α^i is a valuation over $\mathsf{vars}(q)$ such that $\alpha^i(q) \subseteq \mathbf{r}'^i$;
3. $\alpha^i(\{F_j\}_{j=1}^i) = \beta_b(\{F_j\}_{j=1}^i)$ and $\alpha^i(\vec{z}) = \beta_b(\vec{z})$ (and therefore $\alpha^i(\vec{z}) = \alpha(\vec{z})$);
4. $\alpha^i(w) = \alpha(w)$; and
5. $\mathsf{Reify}(\mathbf{r}'^i) = \{\alpha[Z \cup \{w\}]\}$.

The third condition entails $\{\alpha^i, \beta_b\} \models \mathcal{K}(\{F_j\}_{j=1}^i)$ for all $i \in \{0, 1, \ldots, \ell\}$. From (4), it follows $\{\alpha^i, \beta_b\} \models Z \to \mathsf{key}(F_{i+1})$. Then, from $\alpha^i(\vec{z}) = \beta_b(\vec{z})$, it follows that α^i and β_b agree on all variables of $\mathsf{key}(F_{i+1})$.

The proof of the above claim runs by induction on increasing i. For the basis of the induction, $i = 0$, the desired result holds by choosing $\mathbf{r}'^0 = \mathbf{r}'$ and $\alpha^0 = \alpha$.

For the induction step, $i \to i+1$, the induction hypothesis is that the desired pair $(\mathbf{r}'^i, \alpha^i)$ exists. Since α^i and β_b agree on all variables of $\mathsf{key}(F_{i+1})$, we have that $\alpha^i(F_{i+1})$ and $\beta_b(F_{i+1})$ are key-equal. From $\beta_b(q) \subseteq \mathbf{db}'$, it follows that $\beta_b(F_{i+1}) \in \mathbf{db}'$. Let $\mathbf{r}'^{i+1} = \left(\mathbf{r}'^i \setminus \{\alpha^i(F_{i+1})\}\right) \cup \{\beta_b(F_{i+1})\}$, which is obviously a repair of $\mathbf{db}'$. Since $F_{i+1} \overset{q}{\not\rightsquigarrow} u$ for all $u \in Z \cup \{w\}$, $\mathsf{Reify}(\mathbf{r}'^{i+1}) \subseteq \mathsf{Reify}(\mathbf{r}'^i)$ by [21, Lemma B.1]. Since we assumed that every repair of $\mathbf{db}'$ satisfies q, we have that $\mathsf{Reify}(\mathbf{r}'^{i+1}) \neq \emptyset$, and therefore $\mathsf{Reify}(\mathbf{r}'^{i+1}) = \{\alpha[Z \cup \{w\}]\}$. Hence, there exists a valuation α^{i+1} over $\mathsf{vars}(q)$ such that $\alpha^{i+1}(q) \subseteq \mathbf{r}'^{i+1}$ and $\alpha^{i+1}[Z \cup \{w\}] = \alpha[Z \cup \{w\}]$, that is, $\alpha^{i+1}(\vec{z}) = \alpha(\vec{z})$ and $\alpha^{i+1}(w) = \alpha(w)$. Since $\alpha(\vec{z}) = \beta_b(\vec{z})$, we have $\alpha^{i+1}(\vec{z}) = \beta_b(\vec{z})$. We have thus shown that the pair $(\mathbf{r}'^{i+1}, \alpha^{i+1})$ satisfies items 1, 2, 4, and 5 in the above five-item list; we also have shown the second conjunct of item 3. In the next paragraph, we show that $\alpha^{i+1}(\{F_j\}_{j=1}^{i+1}) = \beta_b(\{F_j\}_{j=1}^{i+1})$, i.e., the first conjunct of item 3.

[2] Here, $\alpha[Z \cup \{w\}]$ is the restriction of α to $Z \cup \{w\}$.

By the induction hypothesis, $\alpha^i(\{F_j\}_{j=1}^i) = \beta_b(\{F_j\}_{j=1}^i)$ and $\alpha^i(q) \subseteq \mathbf{r}'^i$, which implies $\beta_b(\{F_j\}_{j=1}^i) \subseteq \mathbf{r}'^i$. Since $\mathbf{r}'^i$ and $\mathbf{r}'^{i+1}$ include the same set of F_j-facts for every $j \in \{1, \ldots, i\}$, we have $\beta_b(\{F_j\}_{j=1}^i) \subseteq \mathbf{r}'^{i+1}$. Since $\beta_b(F_{i+1}) \in \mathbf{r}'^{i+1}$ by construction, we obtain $\beta_b(\{F_j\}_{j=1}^{i+1}) \subseteq \mathbf{r}'^{i+1}$. Since also $\alpha^{i+1}(\{F_j\}_{j=1}^{i+1}) \subseteq \mathbf{r}'^{i+1}$ (because $\alpha^{i+1}(q) \subseteq \mathbf{r}'^{i+1}$), it is correct to conclude that $\{\beta_b, \alpha^{i+1}\} \models \mathcal{K}(\{F_j\}_{j=1}^{i+1})$ by Lemma 16. We are now ready to show that $\alpha^{i+1}(F_j) = \beta_b(F_j)$ for all $j \in \{1, \ldots, i+1\}$. To this end, pick any $k \in \{1, \ldots, i+1\}$. We have $\mathcal{K}(\{F_j\}_{j=1}^{k-1}) \models Z \to \mathsf{key}(F_k)$ by (4). Since $\{F_j\}_{j=1}^{k-1}$ is a subset of $\{F_j\}_{j=1}^{i+1}$, we have $\{\beta_b, \alpha^{i+1}\} \models \mathcal{K}(\{F_j\}_{j=1}^{k-1})$, and therefore $\{\beta_b, \alpha^{i+1}\} \models Z \to \mathsf{key}(F_k)$. Then, from $\alpha^{i+1}(\vec{z}) = \beta_b(\vec{z})$ (the second conjunct of item 3), it follows that α^{i+1} and β_b agree on all variables of $\mathsf{key}(F_k)$. Since $\alpha^{i+1}(F_k), \beta_b(F_k) \in \mathbf{r}'^{i+1}$, it must be the case that $\alpha^{i+1}(F_k) = \beta_b(F_k)$. This concludes the induction step.

For the pair $(\mathbf{r}'^{\ell}, \alpha^{\ell})$, we have that $\alpha^{\ell}(\{F_j\}_{j=1}^{\ell}) = \beta_b(\{F_j\}_{j=1}^{\ell})$, and therefore, since w occurs in some F_j, $\alpha^{\ell}(w) = \beta_b(w)$. Since also $\alpha^{\ell}(w) = \alpha(w)$, we obtain $\alpha(w) = \beta_b(w)$, a contradiction. We conclude by contradiction that some repair of $\mathbf{db}'$ falsifies q. Thus, the purification step described in the paragraph immediate following (4) does not change the answer to $\mathsf{CERTAINTY}(q)$.

We repeat the "purification" step until it can no longer be applied. Let the final database be $\widehat{\mathbf{db}}$. By the above reasoning, we have that every repair of $\widehat{\mathbf{db}}$ satisfies q if and only if every repair of $\mathbf{db}$ satisfies q. Let $\mathbf{s}$ be the smallest set of N-facts containing $N(\underline{\beta(\vec{z})}, \beta(w))$ for every valuation β over $\mathsf{vars}(q)$ such that $\beta(q) \subseteq \mathbf{db}$. We show that $\mathbf{s}$ is consistent. To this end, let β_1, β_2 be valuations over $\mathsf{vars}(q)$ such that $\beta_1(q), \beta_2(q) \subseteq \mathbf{db}$ and $\beta_1(\vec{z}) = \beta_2(\vec{z})$. If $\beta_1(w) \neq \beta_2(w)$, then a purification step can remove the block containing $\beta_1(F)$, contradicting our assumption that no purification step is applicable on $\widehat{\mathbf{db}}$. We conclude by contradiction that $\beta_1(w) = \beta_2(w)$.

Since N has mode c and $\mathbf{s}$ is consistent, we have that $\widehat{\mathbf{db}} \cup \mathbf{s}$ is a legal database. It can now be easily seen that every repair of $\mathbf{db}$ satisfies q if and only if every repair of $\widehat{\mathbf{db}} \cup \mathbf{s}$ satisfies $q' = q \cup \{N^{\mathsf{c}}(\underline{\vec{z}}, w)\}$.

It remains to be argued that the reduction is in **FO**, i.e., that the result of the repeated "purification" step can be obtained by a single first-order query. Let $\mathsf{vars}(q) = \{x_1, \ldots, x_n\}$. Let $q^*(x_1, \ldots, x_n) := \bigwedge_{G \in q} G$ be the quantifier-free part of the first-order formula expressing the Boolean query q. For every $i \in \{1, \ldots, n\}$, let x_i' be a fresh variable. Let $\vec{u}$ be a sequence of distinct variables such that $\mathsf{vars}(\vec{u}) = \mathsf{vars}(F)$. The following query finds all F-facts whose blocks can be removed:

$$\left\{ \vec{u} \mid \exists^* \left(q^*(x_1, \ldots, x_n) \wedge q^*(x_1', \ldots, x_n') \wedge \left(\bigwedge_{z \in Z} z = z' \right) \wedge w \neq w' \right) \right\},$$

where the existential quantification ranges over all variables not in $\vec{u}$. The F-facts that are to be preserved are not key-equal to a fact in the preceding query and can obviously be computed in **FO**. This concludes the proof of the first item.

Proof of the Second Item Assume that the attack graph of q contains no strong cycle. We will show that the attack graph of q' contains no strong cycle either. By the second item in Definition 8, we can assume an atom $G \in q$ such that $Z \subseteq \mathsf{vars}(G)$. Note that the atom $N^{\mathsf{c}}(\underline{\vec{z}}, w)$ has no outgoing attacks because its mode is c. It is sufficient to show that for every $F, H \in q$, if there exists a witness for $F \overset{q'}{\rightsquigarrow} H$, then there exists a witness for $F \overset{q'}{\rightsquigarrow} H$ that does not contain $N^{\mathsf{c}}(\underline{\vec{z}}, w)$. To this end, assume that a witness for $F \overset{q'}{\rightsquigarrow} H$ contains

$$\cdots F' \overset{u'}{\frown} N^{\mathsf{c}}(\underline{\vec{z}}, w) \overset{u''}{\frown} F'' \cdots, \tag{5}$$

where u' and u'' are distinct variables. We can assume without loss of generality that this is the only occurrence of $N^{\mathsf{c}}(\underline{\vec{z}}, w)$ in the witness. In this case, we have $F \overset{q}{\rightsquigarrow} u'$. If $u', u'' \in Z$, then we can replace $N^{\mathsf{c}}(\underline{\vec{z}}, w)$ with G. So the only nontrivial case is where either $u' = w$ or $u'' = w$ (but not both). Then, it must be the case that $\mathcal{K}(q' \setminus \{F\}) \not\models \mathsf{key}(F) \to w$, and therefore also

$$\mathcal{K}(q \setminus \{F\}) \not\models \mathsf{key}(F) \to w. \tag{6}$$

Since $Z \to w$ is internal to q, there exists a sequential proof for $\mathcal{K}(q) \models Z \to w$ such that no atom in the proof attacks a variable in $Z \cup \{w\}$. Let $J_1, J_2, \ldots, J_\ell$ be a shortest such proof. Because $F \overset{q}{\rightsquigarrow} u'$ and $u' \in Z \cup \{w\}$, it must be that $F \notin \{J_1, \ldots, J_\ell\}$. We can assume that w occurs at a non-primary-key position in J_ℓ. Because of (6), we can assume the existence of a variable $v \in \mathsf{key}(J_\ell)$ such that $\mathcal{K}(q \setminus \{F\}) \not\models \mathsf{key}(F) \to v$. If $v \notin Z$, then there exists $k < \ell$ such that v occurs at a non-primary-key position in J_k. Again, we can assume a variable $v' \in \mathsf{key}(J_k)$ such that $\mathcal{K}(q \setminus \{F\}) \not\models \mathsf{key}(F) \to v'$. By repeating the same reasoning, there exists a sequence

$$\overset{z_{i_0}}{\frown} J_{i_0} \overset{z_{i_1}}{\frown} J_{i_1} \overset{z_{i_2}}{\frown} \cdots \overset{z_{i_m}}{\frown} J_{i_m} \overset{w}{\frown}$$

where $1 \leq i_0 < i_1 < \cdots < i_m = \ell$ such that

- $z_{i_0} \in Z$;
- for all $j \in \{0, \ldots, m\}$, $\mathcal{K}(q \setminus \{F\}) \not\models \mathsf{key}(F) \to z_{i_j}$; and
- for all $j \in \{1, \ldots, m\}$, $z_{i_j} \in \mathsf{vars}(J_{i_{j-1}}) \cap \mathsf{vars}(J_{i_j})$. In particular, $z_{i_j} \in \mathsf{key}(J_{i_j})$.

We can assume $G \in q$ such that $Z \subseteq \mathsf{vars}(G)$. Let $u \in \{u', u''\}$ such that $u \neq w$. Thus, $\{u, w\} = \{u', u''\}$. It can now be easily seen that a witness for $F \overset{q'}{\rightsquigarrow} H$ can be obtained by replacing $N^{\mathsf{c}}(\underline{\vec{z}}, w)$ in (5) with the following sequence or its reverse:

$$\overset{u}{\frown} G \overset{z_{i_0}}{\frown} J_{i_0} \overset{z_{i_1}}{\frown} J_{i_1} \overset{z_{i_2}}{\frown} \cdots \overset{z_{i_m}}{\frown} J_{i_m} \overset{w}{\frown}$$

This concludes the proof of Lemma 17. □

The proof of Lemma 11 is now straightforward.

Proof of Lemma 11 Repeated application of Lemma 17. □

D.2 Proof of Lemma 12

We will use the following helping lemma.

Lemma 18 *Let q be a query in* sjfBCQ *such that q is saturated and the attack graph of q contains no strong cycle. Let $\mathcal{S}$ be an initial strong component in the attack graph of q with $|\mathcal{S}| \geq 2$. For every atom $F \in \mathcal{S}$, there exists an atom $H \in \mathcal{S}$ such that $F \xrightarrow{\mathsf{M}} H$.*

Proof Assume $F \in \mathcal{S}$. Since F belongs to an initial strong component with at least two atoms, there exists $G \in \mathcal{S}$ such that $F \overset{q}{\rightsquigarrow} G$ and the attack is weak. Therefore, $\mathcal{K}(q) \models \mathsf{key}(F) \rightarrow \mathsf{key}(G)$. It follows that $\mathcal{K}(q \setminus \{F\}) \models \mathsf{vars}(F) \rightarrow \mathsf{key}(G)$. Let $\sigma = H_1, H_2, \ldots, H_\ell$ be a sequential proof for $\mathcal{K}(q \setminus \{F\}) \models \mathsf{vars}(F) \rightarrow \mathsf{key}(G)$, where $F \notin \{H_1, \ldots, H_\ell\}$. We can assume without loss of generality that $H_\ell = G$.

Let j be the smallest index in $\{1, \ldots, \ell\}$ such that $H_j \in \mathcal{S}$. Since $H_\ell \in \mathcal{S}$, such an index always exists. Then, $\sigma = H_1, H_2, \ldots, H_{j-1}$ is a sequential proof for $\mathcal{K}(q \setminus \{F\}) \models \mathsf{vars}(F) \rightarrow \mathsf{key}(H_j)$ (observe that this proof may be empty). By our choice of j, for every $i \in \{1, \ldots, j-1\}$, we have $H_i \notin \mathcal{S}$, and hence H_i cannot attack F or H_j (since $\mathcal{S}$ is an initial strong component). It follows that no atom in σ attacks a variable in $\mathsf{vars}(F) \cup \mathsf{key}(H_j)$. Since q is saturated, this implies that $\mathcal{K}(q^{\mathsf{cons}}) \models \mathsf{vars}(F) \rightarrow \mathsf{key}(H_j)$, and so $F \xrightarrow{\mathsf{M}} H_j$. □

The proof of Lemma 12 can now be given.

Proof of Lemma 12 Starting from some atom $F_0 \in \mathcal{S}$, by applying repeatedly Lemma 18, we can create an infinite sequence $F_0 \xrightarrow{\mathsf{M}} F_1 \xrightarrow{\mathsf{M}} F_2 \xrightarrow{\mathsf{M}} \cdots$ such that for every $i \geq 1$, $F_i \in \mathcal{S}$ and $F_i \neq F_{i+1}$. Since the atoms in $\mathcal{S}$ are finitely many, there will exist some i, j such that $i < j$ and $F_i = F_{j+1}$. It follows that the M-graph of q contains a cycle all of whose atoms belong to $\mathcal{S}$. □

E Proofs of Section 9

We will use the following helping lemma.

Lemma 19 *Let q be a query in* sjfBCQ *that has the key-join property. Then, for all $F, G \in q$, if $F \overset{q}{\rightsquigarrow} G$, then there exists a sequence $F_0, F_1, \ldots, F_\ell$ such that $F_0 = F$, $F_\ell = G$, and for all $i \in \{1, 2, \ldots, \ell\}$, $\mathsf{key}(F_i) \subseteq \mathsf{vars}(F_{i-1})$.*

Proof Assume $F \overset{q}{\rightsquigarrow} G$. We can assume a shortest sequence

$$F_0 \overset{x_1}{\frown} F_1 \overset{x_2}{\frown} F_2 \cdots \overset{x_{\ell-1}}{\frown} F_{\ell-1} \overset{x_\ell}{\frown} F_\ell \tag{7}$$

that is a witness for $F \overset{q}{\rightsquigarrow} G$. Clearly, for all $i \in \{0, 1, \ldots, \ell - 1\}$, $\mathsf{vars}(F_i) \cap \mathsf{vars}(F_{i+1}) \neq \emptyset$. Then, since q has the key-join property, for all $i \in \{0, 1, \ldots, \ell - 1\}$, either

1. $\mathsf{vars}(F_i) \cap \mathsf{vars}(F_{i+1}) \in \{\mathsf{key}(F_i), \mathsf{key}(F_{i+1})\}$, or
2. $\mathsf{vars}(F_i) \cap \mathsf{vars}(F_{i+1}) \supseteq \mathsf{key}(F_i) \cup \mathsf{key}(F_{i+1})$.

We show by induction on increasing i that for all $i \in \{1, \ldots, \ell\}$, $\mathsf{key}(F_i) \subseteq \mathsf{vars}(F_{i-1})$.

Induction Basis $i = 1$ From $x_1 \notin {F_0}^{+,q}$, it follows $x_1 \notin \mathsf{key}(F_0)$. It follows that $\mathsf{vars}(F_0) \cap \mathsf{vars}(F_1) \neq \mathsf{key}(F_0)$. Consequently, $\mathsf{vars}(F_0) \cap \mathsf{vars}(F_1)$ includes $\mathsf{key}(F_1)$.

Induction Step $i \to i + 1$ The induction hypothesis is that $\mathsf{key}(F_i) \subseteq \mathsf{vars}(F_{i-1})$. Assume, towards a contradiction, $\mathsf{vars}(F_i) \cap \mathsf{vars}(F_{i+1}) = \mathsf{key}(F_i)$. It follows $x_{i+1} \in \mathsf{vars}(F_{i-1})$. Then the witness (7) can be shortened by replacing the subsequence $F_{i-1} \overset{x_i}{\frown} F_i \overset{x_{i+1}}{\frown} F_{i+1}$ with $F_{i-1} \overset{x_{i+1}}{\frown} F_{i+1}$, contradicting our assumption that no witness for $F \overset{q}{\rightsquigarrow} G$ is shorter than (7). We conclude by contradiction that $\mathsf{vars}(F_i) \cap \mathsf{vars}(F_{i+1}) \neq \mathsf{key}(F_i)$. Consequently, $\mathsf{vars}(F_i) \cap \mathsf{vars}(F_{i+1})$ includes $\mathsf{key}(F_{i+1})$. □

The proof of Theorem 4 can now be given.

Proof of Theorem 4 Assume that q has the key-join property We show that the attack graph of q contains no strong attacks. To this end, assume $F \overset{q}{\rightsquigarrow} G$. The sequence $F_0, F_1, \ldots, F_{\ell-1}$ in the statement of Lemma 19 is a sequential proof for $\mathcal{K}(q) \models \mathsf{key}(F_0) \to \mathsf{key}(F_\ell)$, and therefore the attack $F \overset{q}{\rightsquigarrow} G$ is weak. The result then follows from Theorem 3. □

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley, Boston (1995). http://webdam.inria.fr/Alice/
2. Arenas, M., Bertossi, L.E., Chomicki, J.: Consistent query answers in inconsistent databases. In: ACM PODS, pp. 68–79. https://doi.org/10.1145/303976.303983 (1999)
3. Arenas, M., Bertossi, L.E., Chomicki, J., He, X., Raghavan, V., Spinrad, J.P.: Scalar aggregation in inconsistent databases. Theor. Comput. Sci. **296**(3), 405–434 (2003). https://doi.org/10.1016/S0304-3975(02)00737-5
4. Aspvall, B., Plass, M.F., Tarjan, R.E.: A linear-time algorithm for testing the truth of certain quantified boolean formulas. Inf. Process. Lett. **8**(3), 121–123 (1979). https://doi.org/10.1016/0020-0190(79)90002-4
5. Baader, F., Horrocks, I., Lutz, C., Sattler, U.: An introduction to description logic. Cambridge University Press, Cambridge (2017). http://www.cambridge.org/de/academic/subjects/computer-science/knowledge-management-databases-and-data-mining/introduction-description-logic?format=PB#17zVGeWD2TZUeu6s.97
6. Barceló, P., Fontaine, G.: On the data complexity of consistent query answering over graph databases. J. Comput. Syst. Sci. **88**, 164–194 (2017). https://doi.org/10.1016/j.jcss.2017.03.015
7. Bertossi, L.E.: Database repairing and consistent query answering. Synthesis lectures on data management. Morgan & Claypool Publishers, San Rafael (2011)

8. Bertossi, L.E.: Database repairs and consistent query answering: Origins and further developments. In: Suciu, D., Skritek, S., Koch, C. (eds.) Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019, pp. 48–58. ACM (2019). https://doi.org/10.1145/3294052.3322190
9. Bienvenu, M., Bourgaux, C.: Inconsistency-tolerant querying of description logic knowledge bases. In: Pan, J.Z., Calvanese, D., Eiter, T., Horrocks, I., Kifer, M., Lin, F., Zhao, Y. (eds.) Reasoning Web: Logical foundation of knowledge graph construction and query answering - 12th International Summer School 2016, Aberdeen, UK, September 5-9, 2016, Tutorial lectures, Lecture notes in computer science, vol. 9885, pp. 156–202. Springer (2016). https://doi.org/10.1007/978-3-319-49493-7_5
10. Bulatov, A.A.: Complexity of conservative constraint satisfaction problems. ACM Trans. Comput. Log. **12**(4), 24:1–24:66 (2011). https://doi.org/10.1145/1970398.1970400
11. Dixit, A.A., Kolaitis, P.G.: A SAT-based system for consistent query answering. In: Janota, M., Lynce, I. (eds.) Theory and Applications of Satisfiability Testing - SAT 2019 - 22nd International Conference, SAT 2019, Lisbon, Portugal, July 9-12, 2019, Proceedings, Lecture Notes in Computer Science, vol. 11628, pp. 117–135. Springer (2019). https://doi.org/10.1007/978-3-030-24258-9_8
12. Egri, L., Larose, B., Tesson, P.: Symmetric Datalog and constraint satisfaction problems in Logspace. In: LICS, pp. 193–202. https://doi.org/10.1109/LICS.2007.47 (2007)
13. Fontaine, G.: Why is it hard to obtain a dichotomy for consistent query answering? ACM Trans. Comput. Log. **16**(1), 7:1–7:24 (2015). https://doi.org/10.1145/2699912
14. Fuxman, A., Miller, R.J.: First-order query rewriting for inconsistent databases. In: ICDT, pp. 337–351 (2005). https://doi.org/10.1007/978-3-540-30570-5_23
15. Fuxman, A., Miller, R.J.: First-order query rewriting for inconsistent databases. J. Comput. Syst. Sci. **73**(4), 610–635 (2007). https://doi.org/10.1016/j.jcss.2006.10.013
16. Grädel, E., Kolaitis, P.G., Libkin, L., Marx, M., Spencer, J., Vardi, M.Y., Venema, Y., Weinstein, S.: Finite model theory and its applications. Texts in theoretical computer science. An EATCS series springer. https://doi.org/10.1007/3-540-68804-8 (2007)
17. Greco, S., Pijcke, F., Wijsen, J.: Certain query answering in partially consistent databases. PVLDB **7**(5), 353–364 (2014). http://www.vldb.org/pvldb/vol7/p353-greco.pdf
18. Grohe, M., Schwentick, T.: Locality of order-invariant first-order formulas. ACM Trans. Comput. Log. **1**(1), 112–130 (2000). https://doi.org/10.1145/343369.343386
19. Kolaitis, P.G., Pema, E., Tan, W.: Efficient querying of inconsistent databases with binary integer programming. PVLDB **6**(6), 397–408 (2013). http://www.vldb.org/pvldb/vol6/p397-tan.pdf
20. Koutris, P., Wijsen, J.: The data complexity of consistent query answering for self-join-free conjunctive queries under primary key constraints. In: PODS, pp. 17–29 (2015). https://doi.org/10.1145/2745754.2745769
21. Koutris, P., Wijsen, J.: Consistent query answering for self-join-free conjunctive queries under primary key constraints. ACM Trans. Database Syst. **42**(2), 9:1–9:45 (2017). https://doi.org/10.1145/3068334
22. Koutris, P., Wijsen, J.: Consistent query answering for primary keys and conjunctive queries with negated atoms. In: PODS, pp. 209–224 (2018). https://doi.org/10.1145/3196959.3196982
23. Koutris, P., Wijsen, J.: Consistent query answering for primary keys in logspace. In: Barceló, P., Calautti, M. (eds.) 22nd International Conference on Database Theory, ICDT 2019, March 26-28, 2019, Lisbon, Portugal, LIPIcs, vol. 127, pp. 23:1–23:19 (2019). https://doi.org/10.4230/LIPIcs.ICDT.2019.23. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik
24. Lembo, D., Lenzerini, M., Rosati, R., Ruzzi, M., Savo, D.F.: Inconsistency-tolerant query answering in ontology-based data access. J. Web Sem. **33**, 3–29 (2015). https://doi.org/10.1016/j.websem.2015.04.002
25. Libkin, L.: Elements of finite model theory. Texts in theoretical computer science. An EATCS series springer. https://doi.org/10.1007/978-3-662-07003-1 (2004)
26. Lincoln, A., Williams, V.V., Williams, R.R.: Tight hardness for shortest cycles and paths in sparse graphs. In: ACM-SIAM SODA, pp. 1236–1252 (2018). https://doi.org/10.1137/1.9781611975031.80
27. Lutz, C., Wolter, F.: On the relationship between consistent query answering and constraint satisfaction problems. In: ICDT, pp. 363–379 (2015). https://doi.org/10.4230/LIPIcs.ICDT.2015.363
28. Marileo, M.C., Bertossi, L.E.: The consistency extractor system: Answer set programs for consistent query answering in databases. Data Knowl. Eng. **69**(6), 545–572 (2010). https://doi.org/10.1016/j.datak.2010.01.005
29. Maslowski, D., Wijsen, J.: A dichotomy in the complexity of counting database repairs. J. Comput. Syst. Sci. **79**(6), 958–983 (2013). https://doi.org/10.1016/j.jcss.2013.01.011

30. Maslowski, D., Wijsen, J.: Counting database repairs that satisfy conjunctive queries with self-joins. In: ICDT, pp. 155–164 (2014). https://doi.org/10.5441/002/icdt.2014.18
31. Pijcke, F.: Theoretical and practical methods for consistent query answering in the relational data model. Ph.D. thesis, University of Mons (2018)
32. Przymus, P., Boniewicz, A., Burzanska, M., Stencel, K.: Recursive query facilities in relational databases: a survey. In: FGIT, pp. 89–99 (2010). https://doi.org/10.1007/978-3-642-17622-7_10
33. Reingold, O.: Undirected connectivity in log-space. J. ACM **55**(4), 17:1–17:24 (2008). https://doi.org/10.1145/1391289.1391291
34. Wijsen, J.: On the First-order expressibility of computing certain answers to conjunctive queries over uncertain databases. In: PODS, pp. 179–190 (2010). https://doi.org/10.1145/1807085.1807111
35. Wijsen, J.: Certain conjunctive query answering in first-order logic. ACM Trans. Database Syst. **37**(2), 9:1–9:35 (2012). https://doi.org/10.1145/2188349.2188351
36. Wijsen, J.: A survey of the data complexity of consistent query answering under key constraints. In: FoIKS, pp. 62–78 (2014). https://doi.org/10.1007/978-3-319-04939-7_2
37. Wijsen, J.: Foundations of query answering on inconsistent databases. SIGMOD Rec. **48**(3), 6–16 (2019). https://doi.org/10.1145/3377391.3377393