Certain Conjunctive Query Answering in First-Order Logic

JEF WIJSEN, Université de Mons

Primary key violations provide a natural means for modeling uncertainty in the relational data model. A repair (or possible world) of a database is then obtained by selecting a maximal number of tuples without ever selecting two distinct tuples that have the same primary key value. For a Boolean query q, the problem CERTAINTY(q) takes as input a database **db** and asks whether q evaluates to true on every repair of **db**. We are interested in determining queries q for which CERTAINTY(q) is first-order expressible (and hence in the low complexity class $\mathbf{AC}^0$). For queries q in the class of conjunctive queries without self-join, we provide a necessary syntactic condition for first-order expressibility of CERTAINTY(q). For acyclic queries (in the sense of Beeri et al. [1983]), this necessary condition is also a sufficient condition. So we obtain a decision procedure for first-order expressibility of CERTAINTY(q) when q is acyclic and without self-join. We also show that if CERTAINTY(q) is first-order expressible, its first-order definition, commonly called certain first-order rewriting, can be constructed in a rather straightforward way.

Categories and Subject Descriptors: H.2.3 [**Database Management**]: Languages—*Query languages*; H.2.4 [**Database Management**]: Systems—*Relational databases*

General Terms: Theory, Algorithms

Additional Key Words and Phrases: Conjunctive queries, consistent query answering, first-order expressibility, primary keys

ACM Reference Format:
Wijsen, J. 2012. Certain conjunctive query answering in first-order logic. ACM Trans. Datab. Syst. 37, 2, Article 9 (May 2012), 35 pages.
DOI = 10.1145/2188349.2188351 http://doi.acm.org/10.1145/2188349.2188351

1. INTRODUCTION

Uncertainty is inherent in many database applications and can be modeled in the relational data model by means of relations that violate their primary key constraint. Such uncertainty is not necessarily a bad thing. In planning databases, for example, primary key violations can represent different alternatives. In the conference planning database of Figure 1, where primary keys are underlined, the exact town of VLDB 2016 is still uncertain (it can be Milan or Naples). On the other hand, even though we do not know the exact town, we can say that VLDB will be held in Italy. Uncertainty also arises as an inconvenient but inescapable consequence of data integration and data exchange. The relation **T** in Figure 1 combines data from two different sources, each providing a different country for the city of Ferrara. Existing chase-based data exchange frameworks [Fagin et al. 2005] provide no solution in this case, because the chase will fail when it tries to identify Italy and Greece.

This work is a revised and extended version of Wijsen [2010a].

Author's address: J. Wijsen, Université de Mons, 20 Place du Parc, 7000 Mons, Belgium; email: jef.wijsen@umons.ac.be.

DOI 10.1145/2188349.2188351 http://doi.acm.org/10.1145/2188349.2188351

R	Conf	Year	Town
	VLDB	2016	Milan
	VLDB	2016	Naples

T	Town	Country
	Milan	Italy
	Naples	Italy
	Ferrara	Italy
	Ferrara	Greece

Fig. 1. Conference planning database.

Uncertainty by primary key violations gives rise to (exponentially many) "possible worlds," which we will call *repairs*: every repair is obtained by selecting a maximal number of tuples from each relation without ever selecting two distinct tuples that agree on their primary key. A Boolean query is then certain if it evaluates to true on every repair. Our example database has four repairs, each satisfying the Boolean conjunctive query:

$$q_1 = \exists x \exists y \big(\mathbf{R}(\underline{\text{'VLDB'}, x}, y) \wedge \mathbf{T}(\underline{y}, \text{'Italy'})\big),$$

stating that VLDB will be organized in Italy in some year.

In this article, we are interested in determining the certainty of Boolean queries by means of a technique known as certain first-order rewriting. To see that q_1 is true in every repair, there is actually no need to evaluate q_1 on all repairs. It suffices to check that the following first-order sentence φ_1 evaluates to true on the original database.

$$\begin{aligned}\varphi_1 \;=\; \exists x \exists y \Bigg(& \mathbf{R}(\underline{\text{'VLDB'}, x}, y) \\ & \wedge \forall y \Bigg(\mathbf{R}(\underline{\text{'VLDB'}, x}, y) \rightarrow \Big(\mathbf{T}(\underline{y}, \text{'Italy'}) \\ & \wedge \forall z \big(\mathbf{T}(\underline{y}, z) \rightarrow z = \text{'Italy'} \big) \Big) \Bigg) \Bigg)\end{aligned}$$

Formally, a certain first-order rewriting for a Boolean query q is a first-order sentence φ such that for every database **db**, q evaluates to true on every repair of **db** if and only if φ evaluates to true on **db**.

The interest of first-order rewriting is evident: to know whether q is true in every repair, it suffices to execute φ once on the original database. Since φ is first-order, it can be encoded in SQL and executed in polynomial-time data complexity using standard database technology. An alternative (but equivalent) way for defining first-order rewriting makes use of the following set, where q is any Boolean query:

$$\mathsf{CERTAINTY}(q) = \{\mathbf{db} \mid q \text{ evaluates to true on every repair of database } \mathbf{db}\}.$$

Saying that q has a certain first-order rewriting is then equivalent to saying that the set CERTAINTY(q) is first-order expressible.

We study the decidability of first-order expressibility of CERTAINTY(q) when q is a conjunctive query without self-join (i.e., without repeated relation names). This issue is not new. Fuxman and Miller [2005, 2007] were the first ones to focus on CERTAINTY(q) for conjunctive queries q without self-join. Their work brings up the following two interesting problems, where q ranges over the class of Boolean conjunctive queries without self-join.

(1) Find a syntactic characterization (in terms of the syntax of q) of the frontier between first-order expressible and not first-order expressible cases of CERTAINTY(q).
(2) Find a syntactic characterization of the frontier between tractable and intractable cases of CERTAINTY(q).

Some partial results are known. The problem $\mathsf{CERTAINTY}(q)$ is obviously in **coNP** for first-order queries q, and it is known that $\mathsf{CERTAINTY}(q_2)$ is **coNP**-complete for $q_2 = \exists x \exists y \exists z (R(\underline{x}, z) \wedge S(\underline{y}, z))$ [Chomicki and Marcinkowski 2005; Fuxman and Miller 2007]. Recently, Wijsen [2010b] pointed out that the two frontiers in (1) and (2) do not coincide, by providing a query q such that $\mathsf{CERTAINTY}(q)$ is in **P** but not first-order expressible. In this article, we present a significant breakthrough: we solve (1) for queries q that are acyclic in the sense of Beeri et al. [1983]. Moreover, our proofs are constructive: if $\mathsf{CERTAINTY}(q)$ is first-order expressible, then a certain first-order rewriting for q can be effectively constructed.

The set of acyclic conjunctive queries without self-join is a large class of queries of practical interest. We briefly discuss the remaining syntactic restrictions. First, the restriction to queries without self-join is not uncommon in uncertain [Fuxman and Miller 2007] and probabilistic databases [Dalvi and Suciu 2007]. Moreover, it is known that self-joins quickly result in first-order inexpressibility [Wijsen 2009b]. Second, the acyclicity restriction implies the existence of join trees, which are helpful in the technical development. Not all our results, however, require acyclicity.

The article is organized as follows. Section 2 introduces the mathematical concepts and terminology. Section 3 discusses related work. Section 4 introduces the construct of *attack graph*, a novel tool for studying first-order (in)expressibility. Section 5 provides a sufficient condition under which a conjunctive query q, without self-join, has no certain first-order rewriting. The main contribution of this article concerns acyclic conjunctive queries. It is shown in Section 6 that an acyclic conjunctive query has a unique attack graph, even if it has multiple join trees. Sections 7 and 8 derive a sufficient and necessary condition for first-order expressibility of $\mathsf{CERTAINTY}(q)$ when q is acyclic and without self-join. Section 9 settles the complexity of determining first-order expressibility of $\mathsf{CERTAINTY}(q)$. Section 10 concludes the article. The appendix contains some technical helping lemmas and some proofs.

2. NOTATIONS AND TERMINOLOGY

We assume disjoint sets of *variables* and *constants*. Variables and constants are *symbols*. If $\vec{x}$ is a sequence of symbols, then $\mathsf{vars}(\vec{x})$ is the set of variables that occur in $\vec{x}$.

Let U be a set of variables. A *valuation over* U is a total mapping θ from U to the set of constants. Such valuation θ is often extended to be the identity on constants and on variables not in U.

Key-equal atoms. Every *relation name* R has a fixed *signature*, which is a pair $[n, k]$ with $n \geq k \geq 1$: the integer n is the *arity* of the relation name and $\{1, 2, \ldots, k\}$ is the *primary key*. The relation name R is *all-key* if $n = k$. If R is a relation name with signature $[n, k]$, then $R(s_1, \ldots, s_n)$ is an *R-atom* (or simply atom), where each s_i is a constant or a variable ($1 \leq i \leq n$). Such 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 atom is *ground* if it contains no variables. Two ground atoms $R_1(\underline{\vec{a}_1}, \vec{b}_1)$, $R_2(\underline{\vec{a}_2}, \vec{b}_2)$ are *key-equal* if $R_1 = R_2$ and $\vec{a}_1 = \vec{a}_2$.

Database and repair. 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 ground atoms using only the relation names of the schema. Importantly, a database can contain distinct, key-equal atoms. Intuitively, if a database contains distinct, key-equal atoms A and B, then only one of A or B can be true, but we do not know which one. In this respect, the database contains uncertainty. A database **db** is *consistent* if it does not contain two distinct atoms that are key-equal. A *repair* of a database **db** is a maximal (under set inclusion) consistent subset of **db**.

Boolean conjunctive query. 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.[1] This set represents the first-order sentence $\exists u_1 \ldots \exists u_k \big(R_1(\underline{\vec{x}_1}, \vec{y}_1) \wedge \cdots \wedge R_n(\underline{\vec{x}_n}, \vec{y}_n)\big)$ where $u_1, \ldots, u_k$ are all the variables that occur in $\vec{x}_1\vec{y}_1 \ldots \vec{x}_n\vec{y}_n$. This query q is *satisfied* by a database $\mathbf{db}$, denoted $\mathbf{db} \models q$, if there exists a valuation θ over $\mathsf{vars}(\vec{x}_1\vec{y}_1 \ldots \vec{x}_n\vec{y}_n)$ such that for each $i \in \{1, \ldots, n\}$, $R_i(\underline{\theta(\vec{x}_i)}, \theta(\vec{y}_i)) \in \mathbf{db}$. We say that q has a *self-join* if some relation name occurs more than once in q.

The restriction to Boolean queries simplifies the technical treatment, but is not fundamental; Section 8 explains how to deal with nonBoolean queries. Since every relation name has a fixed signature, relevant primary key constraints are implicitly present in all queries; moreover, primary keys will be underlined.

Certain query answering. Let q be a Boolean conjunctive query and $\mathbf{db}$ a database. We write $\mathbf{db} \models_{\mathsf{sure}} q$ if for every repair $\mathbf{rep}$ of $\mathbf{db}$, we have $\mathbf{rep} \models q$. Given a Boolean conjunctive query q, $\mathsf{CERTAINTY}(q)$ is (the complexity of) the following set.

$$\mathsf{CERTAINTY}(q) = \{\mathbf{db} \mid \mathbf{db} \text{ is a database and } \mathbf{db} \models_{\mathsf{sure}} q\}$$

$\mathsf{CERTAINTY}(q)$ is said to be *first-order expressible* if there exists a first-order sentence φ such that for every database $\mathbf{db}$, $\mathbf{db} \in \mathsf{CERTAINTY}(q)$ if and only if $\mathbf{db} \models \varphi$. The formula φ, if it exists, is called a *certain first-order rewriting* for q.

Notational conventions. We will use letters A, B, C for ground atoms in a database, and F, G, H, J for atoms appearing in a query. For $F = R(\underline{\vec{x}}, \vec{y})$, we denote by $\mathsf{KVars}(F)$ the set of variables that occur in $\vec{x}$, and by $\mathsf{Vars}(F)$ the set of variables that occur in F, that is, $\mathsf{KVars}(F) = \mathsf{vars}(\vec{x})$ and $\mathsf{Vars}(F) = \mathsf{vars}(\vec{x}) \cup \mathsf{vars}(\vec{y})$.

Acyclic conjunctive queries. An *intersection tree* for a conjunctive query q is an edge-labeled undirected tree whose vertices are the atoms of q; an edge between atoms F and G is labeled by the (possibly empty) set $\mathsf{Vars}(F) \cap \mathsf{Vars}(G)$. An intersection tree for q is called a *join tree* for q if it satisfies the following condition.

Connectedness Condition. Whenever the same variable x occurs in two atoms F and G, then x occurs in each atom on the unique path linking F and G.

The term Connectedness Condition appears in Gottlob et al. [2002] and refers to the fact that the set of vertices in which x occurs induces a connected subtree. A conjunctive query q is *acyclic* if it has a join tree. The notions of join tree and acyclicity are standard [Beeri et al. 1983]. The weaker notion of intersection tree is not in the standard literature, but is derived from the notion of intersection graph defined in Maier [1983, page 453]. The symbol τ will be used for join trees, and ρ for intersection trees. We write $F \overset{L}{\frown} G$ to denote an edge between F and G with label L.

A join tree is shown in Figure 2 (left). The query $\{R_0(\underline{y, z}, u), R_1(\underline{x}, y), R_2(\underline{z}, x, u)\}$ is cyclic and hence has no join tree. An intersection tree for that query is shown in Figure 8 (left).

3. RELATED WORK

Certain (or consistent) query rewriting goes back to Arenas et al. [1999]. Fuxman and Miller [2007] were the first ones to focus on certain first-order rewriting of conjunctive queries under primary key constraints, with applications in the ConQuer system [Fuxman et al. 2005]. They defined a class of conjunctive queries without self-join, called $\mathcal{C}_{forest}$, such that every query in that class has a certain first-order rewriting. At the same time, however, they recognized that the query $q_3 = \exists x \exists y(R(\underline{x}, y) \wedge S(\underline{x}, y))$ is

[1] Up to Section 8, all queries are understood to be Boolean and quantifiers are omitted.

not in $\mathcal{C}_{forest}$ and yet has a certain first-order rewriting. A larger class of conjunctive queries (including q_3) that admit certain first-order rewriting was presented in Wijsen [2009a], where it was also shown that a join of two distinct relations outside that class cannot have a certain first-order rewriting (but no such inexpressibility result was obtained for joins of three or more relations). The generalization to acyclic joins of any number of relations was made in Wijsen [2010a].

The current article focuses on conjunctive queries that are acyclic in the sense of Beeri et al. [1983]. It follows from the proof of Corollary 5 in Wijsen [2009b] that acyclicity is also implicit in the class $\mathcal{C}_{forest}$.

Relatively little is known about CERTAINTY(q) for conjunctive queries q that are cyclic and/or contain self-joins. For the query $q_4 = \exists x \exists y \big(R(\underline{x}, y) \wedge R(\underline{y}, a)\big)$, with self-join, it is known that CERTAINTY(q_4) is in $\mathbf{P}$ but not first-order expressible [Wijsen 2009b]. Fuxman and Miller [2007] claim **coNP**-hardness of CERTAINTY(q) for queries q of the following form, for some $m \geq 2$.

$$\exists x_1 \ldots \exists x_m \big(R_1(\underline{x_1}, x_2) \wedge R_2(\underline{x_2}, x_3) \wedge \cdots \wedge R_{m-1}(\underline{x_{m-1}}, x_m) \wedge R_m(\underline{x_m}, x_1)\big)$$

Queries of this form are cyclic if $m \geq 3$. Unfortunately, these claims are based on incorrect arguments, as remarked by Wijsen [2010b], who showed that for $q_5 = \exists x_1 \exists x_2 \big(R_1(\underline{x_1}, x_2) \wedge R_2(\underline{x_2}, x_1)\big)$ (that is, for $m = 2$), it is the case that CERTAINTY(q_5) is in $\mathbf{P}$, albeit not first-order expressible.

In Wijsen [2009b], Wijsen defined a semantic class $\mathcal{C}_{rooted}$ of conjunctive queries, and showed first-order expressibility of CERTAINTY(q) for all $q \in \mathcal{C}_{rooted}$. Queries in $\mathcal{C}_{rooted}$ can be cyclic and contain self-joins. However, no membership test for $\mathcal{C}_{rooted}$ is known. From the results in the current article, it follows that if an acyclic conjunctive query without self-join has a certain first-order rewriting, then it belongs to $\mathcal{C}_{rooted}$.

It is an open conjecture that for every conjunctive query q, without self-join, it is the case that CERTAINTY(q) is in $\mathbf{P}$ or **coNP**-complete. For queries with exactly two atoms, such dichotomy was recently shown true [Kolaitis and Pema 2012].

Maslowski and Wijsen [2011] have studied the complexity of the counting variant of CERTAINTY(q), denoted ♮CERTAINTY(q). Given a database **db**, the problem ♮CERTAINTY(q) asks to determine the exact number of repairs of **db** that satisfy q. They showed that the class of conjunctive queries q without self-join exhibits a dichotomy: ♮CERTAINTY(q) is in $\mathbf{P}$ or ♮$\mathbf{P}$-complete. The problem ♮CERTAINTY(q) is closely related to query answering in probabilistic data models [Andritsos et al. 2006; Huang et al. 2009; Dalvi et al. 2009]. From the probabilistic database angle, our uncertain databases are a restricted case of *block-independent-disjoint* probabilistic databases [Dalvi et al. 2009, 2011]. A *block* in a database **db** is a maximal subset of key-equal atoms. If $\{R(\underline{\vec{a}}, \vec{b}_1), \ldots, R(\underline{\vec{a}}, \vec{b}_n)\}$ is a block of size n, then every atom of the block has a probability of $1/n$ to be selected in a repair of **db**. Every repair is a possible world, and all these worlds have the same probability. Research in probabilistic databases has studied the complexity of computing the marginal probability of Boolean conjunctive queries. The problem CERTAINTY(q) is to decide whether this marginal probability is equal to 1.

The problem of certain conjunctive query answering under primary keys can be extended in several ways. Grieco et al. [2005] have studied certain query answering under both key and exclusion dependencies. Lembo et al. [2006] have studied certain first-order order rewriting of unions of conjunctive queries under key dependencies.

4. ATTACK GRAPH

We compute for each intersection tree a new graph, called attack graph. Attack graphs will be the tool used for deciding the existence of certain first-order rewritings.

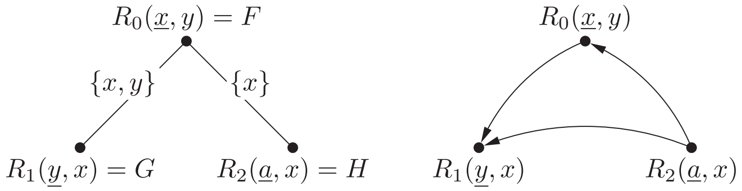


Fig. 2. Join tree τ_6 (left) and attack graph (right) for query q_6. The attack graph is acyclic.

Every atom F in a query q gives rise to a functional dependency among the variables that occur in F. For example, $R(\underline{x, y}, z)$ gives rise to $\{x, y\} \rightarrow \{z\}$. The set $\mathcal{K}(q)$ defined next collects all functional dependencies that arise in atoms of q.

Definition 4.1. Let q be a Boolean conjunctive query. We define $\mathcal{K}(q)$ as the following set of functional dependencies.

$$\mathcal{K}(q) = \{\mathsf{KVars}(F) \rightarrow \mathsf{Vars}(F) \mid F \in q\}$$

Example 4.2. Let q_6 denote the query whose join tree τ_6 is shown in Figure 2 (left). Then, $\mathcal{K}(q_6)$ contains $\{x\} \rightarrow \{x, y\}$, $\{y\} \rightarrow \{x, y\}$, and $\{\} \rightarrow \{x\}$. The latter functional dependency arises in the atom $R_2(\underline{a}, x)$ whose primary key value contains no variables.

To understand the statement of the following lemma, notice that a valuation over a finite set of variables can be regarded as a tuple by treating variables as attributes. For example, let θ be the valuation (or tuple) over $\{x, y, z\}$ defined by $\theta = \{x \mapsto a, y \mapsto b, z \mapsto c\}$. Let $\mu = \{x \mapsto a, y \mapsto b, z \mapsto d\}$. Then, $\{\theta, \mu\}$ satisfies the functional dependency $x \rightarrow y$ because θ and μ agree on y. On the other hand, $\{\theta, \mu\}$ falsifies $x \rightarrow z$ because θ and μ agree on x but disagree on z.

LEMMA 4.3. *Let q be a Boolean conjunctive query (possibly with self-joins). Let U be the set of variables that occur in q. Let* **db** *be a consistent database. If θ, μ are valuations over U such that $\theta(q), \mu(q) \subseteq$* **db***, then $\{\theta, \mu\} \models \mathcal{K}(q)$.*

PROOF. Let $X \rightarrow Y$ be an arbitrary functional dependency in $\mathcal{K}(q)$. We can assume an atom $R(\underline{\vec{x}}, \vec{y})$ of q such that $X = \mathsf{vars}(\vec{x})$ and $Y = \mathsf{vars}(\vec{x}) \cup \mathsf{vars}(\vec{y})$. Assume $\theta[X] = \mu[X]$. We obviously have $\theta(\vec{x}) = \mu(\vec{x})$. From $\theta(q), \mu(q) \subseteq \mathbf{db}$, it follows that $R(\underline{\theta(\vec{x})}, \theta(\vec{y})), R(\underline{\mu(\vec{x})}, \mu(\vec{y})) \in \mathbf{db}$. Since $\theta(\vec{x}) = \mu(\vec{x})$ and since **db** contains no two distinct key-equal atoms, we have $\theta(\vec{y}) = \mu(\vec{y})$. It follows $\theta[Y] = \mu[Y]$. □

Concerning the following definition, recall from relational database theory [Ullman 1988, page 387] that if Σ is a set of functional dependencies over a set U of attributes and $X \subseteq U$, then the attribute closure of X (with respect to Σ) is the set $\{A \in U \mid \Sigma \models X \rightarrow A\}$. We say that X is closed if X is equal to the closure of X.

Definition 4.4. Let q be a Boolean conjunctive query. Let U be the set of variables that occur in q. For every $F \in q$, we define

$$F^{+,q} = \{x \in U \mid \mathcal{K}(q \setminus \{F\}) \models \mathsf{KVars}(F) \rightarrow x\}.$$

In words, $F^{+,q}$ is the attribute closure of the set $\mathsf{KVars}(F)$ with respect to the set of functional dependencies that arise in the atoms of $q \setminus \{F\}$. Note that variables play the role of attributes in our framework.

We now define attack graphs. Every intersection tree has a unique attack graph. The vertices of an intersection tree and its attack graph are the same. Attack graphs, unlike intersection trees, are directed graphs. The construct of attack graph will turn out to be a powerful tool for characterizing first-order expressibility of $\mathsf{CERTAINTY}(q)$.

In particular, we will show that the following properties hold for all Boolean conjunctive queries q without self-join

- —If ρ is an intersection tree for q such that the attack graph of ρ has a cycle with exactly two vertices, then $\mathsf{CERTAINTY}(q)$ is not first-order expressible. This is Theorem 5.1.
- —If q is acyclic, then all join trees for q have the same attack graph. This is Theorem 6.1. It has the important consequence that we can talk about the attack graph of an acyclic conjunctive query q, which is obtained by computing the attack graph of any join tree for q. Proposition 6.4 provides a basic intuition underlying the use of attack graphs.
- —If q is acyclic and the attack graph of q has a cycle (of any length), then $\mathsf{CERTAINTY}(q)$ is not first-order expressible. This is Theorem 7.4.
- —If q is acyclic and the attack graph of q is acyclic, then $\mathsf{CERTAINTY}(q)$ is first-order expressible. In this case, a certain first-order rewriting for q can be constructed from a topological sorting of the attack graph. This is Theorem 8.13.

Definition 4.5. Let ρ be an intersection tree for Boolean conjunctive query q. The *attack graph* of ρ is a directed graph whose vertices are the atoms of q. There is a directed edge from F to G if F, G are distinct atoms such that for every label L on the unique path that links F and G in ρ, we have $L \not\subseteq F^{+,q}$.

We write $F \overset{\rho}{\rightsquigarrow} G$ if the attack graph of ρ contains a directed edge from F to G. If $F \overset{\rho}{\rightsquigarrow} G$, we say that F *attacks* G (or that G is attacked by F). A *cycle of size n* in the attack graph is then a sequence of edges $F_0 \overset{\rho}{\rightsquigarrow} F_1 \overset{\rho}{\rightsquigarrow} F_2 \ldots \overset{\rho}{\rightsquigarrow} F_{n-1} \overset{\rho}{\rightsquigarrow} F_0$.

Example 4.6. Consider again join tree τ_6 for query q_6 in Figure 2 (left). To shorten notation, let $F = R_0(\underline{x}, y)$, $G = R_1(\underline{y}, x)$, and $H = R_2(\underline{a}, x)$, as indicated in Figure 2 (left). The attack graph of τ_6 is shown in Figure 2 (right) and is computed as follows. We have

$$\begin{aligned}
\mathcal{K}(q_6 \setminus \{F\}) &= \{\{y\} \rightarrow \{x, y\}, \{\} \rightarrow \{x\}\}\\
\mathcal{K}(q_6 \setminus \{G\}) &= \{\{x\} \rightarrow \{x, y\}, \{\} \rightarrow \{x\}\}\\
\mathcal{K}(q_6 \setminus \{H\}) &= \{\{x\} \rightarrow \{x, y\}, \{y\} \rightarrow \{x, y\}\}.
\end{aligned}$$

We have $\mathsf{KVars}(F) = \{x\}$, which is already closed with respect to $\mathcal{K}(q_6 \setminus \{F\})$. Thus, $F^{+,q_6} = \{x\}$. The path from F to G in the join tree is $F \overset{\{x,y\}}{\frown} G$. Since the label $\{x, y\}$ is not contained in F^{+,q_6}, the attack graph contains a directed edge from F to G, that is, $F \overset{\tau_6}{\rightsquigarrow} G$. The path from F to H in the join tree is $F \overset{\{x\}}{\frown} H$. Since the label $\{x\}$ is contained in F^{+,q_6}, the attack graph contains no directed edge from F to H.

We have $\mathsf{KVars}(G) = \{y\}$ and the closure of $\{y\}$ with respect to $\mathcal{K}(q_6 \setminus \{G\})$ is $\{x, y\}$. Thus, $G^{+,q_6} = \{x, y\}$. The path from G to F in the join tree is $G \overset{\{x,y\}}{\frown} F$. Since the label $\{x, y\}$ is contained in G^{+,q_6}, the attack graph contains no directed edge from G to F. For that same reason, the attack graph contains no directed edge from G to H.

Finally, we have $\mathsf{KVars}(H) = \{\}$, which is already closed with respect to $\mathcal{K}(q_6 \setminus \{H\})$. Thus, $H^{+,q_6} = \{\}$. The path from H to G in the join tree is $H \overset{\{x\}}{\frown} F \overset{\{x,y\}}{\frown} G$. Since no label on that path is contained in H^{+,q_6}, the attack graph contains a directed edge from H to G, that is, $H \overset{\tau_6}{\rightsquigarrow} G$. It is then obvious that the attack graph must also contain a directed edge from H to F, that is, $H \overset{\tau_6}{\rightsquigarrow} F$.

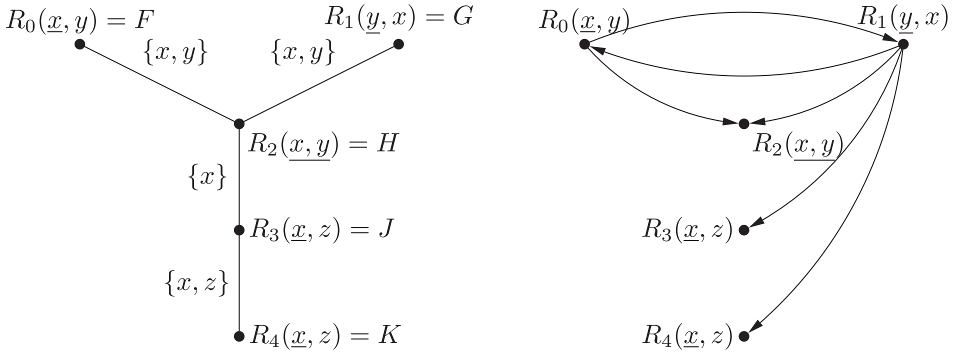


Fig. 3. Join tree τ_7 (left) and attack graph (right) for query q_7. The attack graph contains a cycle.

Example 4.7. Consider join tree τ_7 for query q_7 in Figure 3 (left). Notice that R_2 is all-key. We have

$$\begin{array}{rcl} \mathcal{K}(q_7 \setminus \{F\}) & \equiv & \{y \rightarrow x, x \rightarrow z\} \\ \mathcal{K}(q_7 \setminus \{G\}) & \equiv & \{x \rightarrow y, x \rightarrow z\} \\ \mathcal{K}(q_7 \setminus \{H\}) = \mathcal{K}(q_7 \setminus \{J\}) = \mathcal{K}(q_7 \setminus \{K\}) & \equiv & \{x \rightarrow y, y \rightarrow x, x \rightarrow z\}. \end{array}$$

Consequently,

$$\begin{array}{rcl} F^{+,q_7} & = & \{x, z\} \\ G^{+,q_7} & = & \{y\} \\ H^{+,q_7} & = & \{x, y, z\} \\ J^{+,q_7} & = & \{x, y, z\} \\ K^{+,q_7} & = & \{x, y, z\}. \end{array}$$

Since no edge label is contained in G^{+,q_7}, the atom G attacks every other atom. The complete attack graph is shown in Figure 3 (right). It contains a cycle $F \overset{\tau_7}{\rightsquigarrow} G \overset{\tau_7}{\rightsquigarrow} F$ of size 2. Notice that $F \overset{\tau_7}{\rightsquigarrow} G$ and $G \overset{\tau_7}{\rightsquigarrow} J$, but $F \overset{\tau_7}{\not\rightsquigarrow} J$. So attack graphs need not be transitive.

Definition 4.8. Let q be a Boolean conjunctive query. Let ρ be an intersection tree for q. If F, G are distinct vertices in ρ, then $[F, G]_\rho$ denotes the set of all atoms on the unique path in ρ linking F and G (including F and G).

LEMMA 4.9. *Let q be a Boolean conjunctive query. Let ρ be an intersection tree for q. Let F, G be distinct atoms of q.*

(1) *If $F \overset{\rho}{\rightsquigarrow} G$, then $\mathsf{KVars}(G) \nsubseteq F^{+,q}$.*
(2) *If $F \overset{\rho}{\rightsquigarrow} G$ and $H \in [F, G]_\rho \setminus \{F\}$, then $F \overset{\rho}{\rightsquigarrow} H$.*
(3) *If $F \overset{\rho}{\rightsquigarrow} G$, then $\mathcal{K}(q \setminus [F, G]_\rho) \models \mathsf{KVars}(F) \rightarrow F^{+,q}$.*

PROOF.

(1) Proof by contraposition. Assume $\mathsf{KVars}(G) \subseteq F^{+,q}$, that is, $\mathcal{K}(q \setminus \{F\}) \models \mathsf{KVars}(F) \rightarrow \mathsf{KVars}(G)$. Let L be the last label on the path in ρ from F to G. Since $\mathsf{KVars}(G) \rightarrow \mathsf{Vars}(G)$ belongs to $\mathcal{K}(q \setminus \{F\})$ and $L \subseteq \mathsf{Vars}(G)$, it follows $\mathcal{K}(q \setminus \{F\}) \models \mathsf{KVars}(G) \rightarrow L$. By transitivity, $\mathcal{K}(q \setminus \{F\}) \models \mathsf{KVars}(F) \rightarrow L$. Then $L \subseteq F^{+,q}$, hence $F \overset{\rho}{\not\rightsquigarrow} G$.
(2) Straightforward.

(3) Assume $F \overset{\rho}{\rightsquigarrow} G$. The computation of the attribute closure $F^{+,q}$ by means of a standard algorithm [Abiteboul et al. 1995, page 165] corresponds to constructing a maximal sequence

$$\begin{array}{rcll} \mathsf{KVars}(F) & = & S_0 & H_1 \\ & & S_1 & H_2 \\ & & \vdots & \vdots \\ & & S_{k-1} & H_k \\ & & S_k, & \end{array}$$

where

(a) $S_0 \subsetneq S_1 \subsetneq \cdots \subsetneq S_{k-1} \subsetneq S_k$; and

(b) for every $i \in \{1, 2, \ldots, k\}$,

(i) $H_i \in q \setminus \{F\}$. Thus, $\mathcal{K}(q \setminus \{F\})$ contains the functional dependency $\mathsf{KVars}(H_i) \rightarrow \mathsf{Vars}(H_i)$.

(ii) $\mathsf{KVars}(H_i) \subseteq S_{i-1}$ and $S_i = S_{i-1} \cup \mathsf{Vars}(H_i)$.

Then, $S_k = F^{+,q}$. Assume towards a contradiction $H_i \in [F, G]_\rho$ for some $i \in \{1, 2, \ldots, k\}$. Notice that $\mathsf{KVars}(H_i) \subseteq F^{+,q}$ by the definition of H_i. By item (1) in the current proof, $F \overset{\rho}{\not\rightsquigarrow} H_i$. By item (2) in the current proof, $F \overset{\rho}{\rightsquigarrow} H_i$, a contradiction. We conclude by contradiction that $H_i \notin [F, G]_\rho$. It follows $\mathcal{K}(q \setminus [F, G]_\rho) \models \mathsf{KVars}(F) \rightarrow F^{+,q}$. □

5. INEXPRESSIBILITY RESULT

This section applies to Boolean conjunctive queries without self-join, which can be cyclic or acyclic. We provide a necessary condition for such queries to have a certain first-order rewriting.

The following theorem is proved by using Hanf-locality of first-order logic, which is defined in Chapter 4 of Libkin [2004]. We prefer not to copy-paste these definitions here, because they are lengthy, and the treatment in Libkin [2004] is excellent. The following proof is involved and applies to any conjunctive query without self-join; a particular instance of the proof for the query $\{R(\underline{x}, y), S(\underline{y}, x)\}$ appears in Wijsen [2010b].

THEOREM 5.1. *Let q be a Boolean conjunctive query without self-join. If q has an intersection tree ρ whose attack graph has a cycle of size 2, then $\mathsf{CERTAINTY}(q)$ is not first-order expressible.*

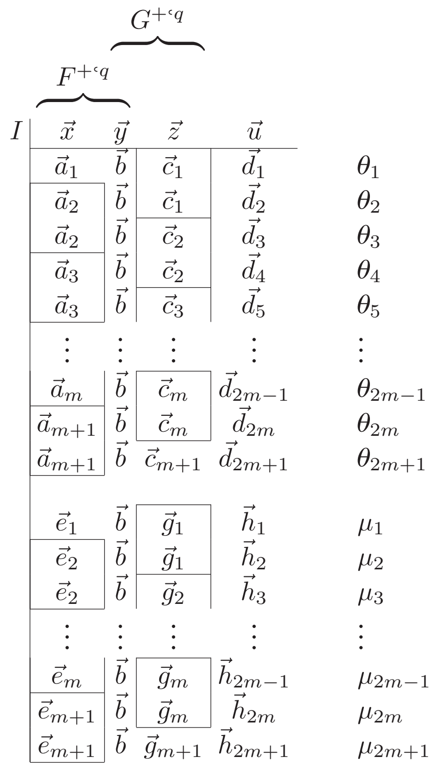
PROOF. Let U be the set of variables that occur in q. Let ρ be an intersection tree for q whose attack graph contains a cycle of size 2. Thus, we can assume two distinct atoms F and G such that $F \overset{\rho}{\rightsquigarrow} G$ and $G \overset{\rho}{\rightsquigarrow} F$.

Assume towards a contradiction that ψ is a first-order sentence such that for every database $\mathbf{db}$, $\mathbf{db} \in \mathsf{CERTAINTY}(q)$ if and only if $\mathbf{db} \models \psi$. From Theorem 4.12 in Libkin [2004], it follows that ψ is Hanf-local. Let $d \geq 0$ be the Hanf-locality rank of ψ.

Choose integer m such that $m > d$. We show the construction of two databases, called $\mathbf{db}_{yes}$ and $\mathbf{db}_{no}$, such that

—$\mathbf{db}_{no} \notin \mathsf{CERTAINTY}(q)$;
—$\mathbf{db}_{yes} \in \mathsf{CERTAINTY}(q)$; and
—$\mathbf{db}_{yes} \models \psi \iff \mathbf{db}_{no} \models \psi$.

This leads to a contradiction that concludes the proof. The following sequences of variables are defined in the construction.



I	$\underbrace{\vec{x} \quad \vec{y}}_{}^{F^{+,q}}$		$\vec{z}$	$\vec{u}$	
	$\vec{x}$	$\vec{y}$ $(G^{+,q}$ spans $\vec{y},\vec{z})$	$\vec{z}$	$\vec{u}$	
	$\vec{a}_1$	$\vec{b}$	$\vec{c}_1$	$\vec{d}_1$	θ_1
	$\vec{a}_2$	$\vec{b}$	$\vec{c}_1$	$\vec{d}_2$	θ_2
	$\vec{a}_2$	$\vec{b}$	$\vec{c}_2$	$\vec{d}_3$	θ_3
	$\vec{a}_3$	$\vec{b}$	$\vec{c}_2$	$\vec{d}_4$	θ_4
	$\vec{a}_3$	$\vec{b}$	$\vec{c}_3$	$\vec{d}_5$	θ_5
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	$\vec{a}_m$	$\vec{b}$	$\vec{c}_m$	$\vec{d}_{2m-1}$	θ_{2m-1}
	$\vec{a}_{m+1}$	$\vec{b}$	$\vec{c}_m$	$\vec{d}_{2m}$	θ_{2m}
	$\vec{a}_{m+1}$	$\vec{b}$	$\vec{c}_{m+1}$	$\vec{d}_{2m+1}$	θ_{2m+1}
	$\vec{e}_1$	$\vec{b}$	$\vec{g}_1$	$\vec{h}_1$	μ_1
	$\vec{e}_2$	$\vec{b}$	$\vec{g}_1$	$\vec{h}_2$	μ_2
	$\vec{e}_2$	$\vec{b}$	$\vec{g}_2$	$\vec{h}_3$	μ_3
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	$\vec{e}_m$	$\vec{b}$	$\vec{g}_m$	$\vec{h}_{2m-1}$	μ_{2m-1}
	$\vec{e}_{m+1}$	$\vec{b}$	$\vec{g}_m$	$\vec{h}_{2m}$	μ_{2m}
	$\vec{e}_{m+1}$	$\vec{b}$	$\vec{g}_{m+1}$	$\vec{h}_{2m+1}$	μ_{2m+1}

Fig. 4. Valuations used in the construction of $\mathbf{db}_{yes}$ and $\mathbf{db}_{no}$.

- —Let $\vec{x}$ be the variables (in some fixed order) of $F^{+,q} \setminus G^{+,q}$. Obviously, $\mathsf{KVars}(F) \subseteq F^{+,q}$. By item (1) in Lemma 4.9, $\mathsf{KVars}(F) \not\subseteq G^{+,q}$. It follows that $\vec{x}$ contains at least one variable.
- —Let $\vec{z}$ be the variables of $G^{+,q} \setminus F^{+,q}$.
- —Let $\vec{y}$ be the variables of $F^{+,q} \cap G^{+,q}$.
- —Let $\vec{u}$ contain all other variables, that is, $\vec{u}$ contains all variables in $(U \setminus F^{+,q}) \setminus G^{+,q}$.

If we treat variables as attributes, a valuation over U is a tuple over U (as illustrated in the paragraph preceding Lemma 4.3). Let I be the set of valuations (or tuples) over U shown in Figure 4. The intended meaning is that distinct letters and/or subscripts denote distinct constants. The boxes in Figure 4 highlight where constants are reused in different valuations. In particular, for all $i, j \in \{1, 2, \ldots, 2m+1\}$, for all $v \in U$, $\theta_i(v) = \mu_j(v) \iff v \in \mathsf{vars}(\vec{y})$. Moreover, for all $i, j \in \{1, 2, \ldots, 2m+1\}$ such that $i < j$, for all $v \in U \setminus \mathsf{vars}(\vec{y})$,

(1) $\theta_i(v) = \theta_j(v)$ if and only if $i + 1 = j$ and one of the following conditions is true:
 —$v \in \mathsf{vars}(\vec{x})$ and i is even; or
 —$v \in \mathsf{vars}(\vec{z})$ and i is odd.

(2) $\mu_i(v) = \mu_j(v)$ if and only if $i + 1 = j$ and one of the following conditions is true:
 —$v \in \mathsf{vars}(\vec{x})$ and i is even; or
 —$v \in \mathsf{vars}(\vec{z})$ and i is odd.

Let

$$\mathbf{db}_\theta = \bigcup \{\theta_i(q) \mid 1 \le i \le 2m+1\}$$

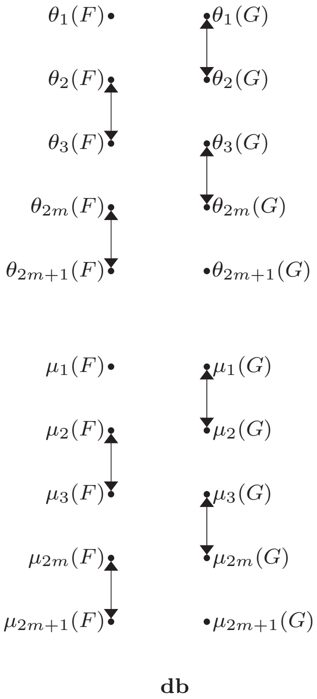


Fig. 5. Case $m = 2$. The figure shows the atoms having the same relation name as F or G in **db**. Conflicting, key-equal atoms are linked by a double-arrowed edge.

$$\mathbf{db}_\mu = \bigcup\{\mu_i(q) \mid 1 \le i \le 2m+1\}$$
$$\mathbf{db} = \mathbf{db}_\theta \cup \mathbf{db}_\mu .$$

Notice that the only constants common to $\mathbf{db}_\theta$ and $\mathbf{db}_\mu$ occur in $\vec{b} = \theta_1(\vec{y})$. We now show that our construction guarantees three remarkable properties. Intuitively, Properties 5.2 and 5.3 imply that the only pairs of key-equal atoms of **db** are the ones depicted in Figure 5. For Property 5.4, say that two distinct atoms $A, B \in \mathbf{db}$ are *joinable* (with respect to q) if $A, B \in \omega(q) \subseteq \mathbf{db}$ for some valuation ω over U. Then, Property 5.4 expresses that two distinct atoms in Figure 5 are joinable only if they appear on the same horizontal line. That is, $\theta_i(F)$ and $\theta_j(G)$ are joinable only if $i = j$; likewise, $\mu_i(F)$ and $\mu_j(G)$ are joinable only if $i = j$; finally, $\theta_i(F)$ and $\mu_j(G)$ are not joinable.

Property 5.2.

(*1*) *for every even number $i \in \{1, 2, \ldots, 2m\}$, $\theta_i(F)$ and $\theta_{i+1}(F)$ are distinct, key-equal atoms;*
(*2*) *for every even number $i \in \{1, 2, \ldots, 2m\}$, $\mu_i(F)$ and $\mu_{i+1}(F)$ are distinct, key-equal atoms;*
(*3*) *for every odd number $i \in \{1, 2, \ldots, 2m\}$, $\theta_i(G)$ and $\theta_{i+1}(G)$ are distinct, key-equal atoms; and*
(*4*) *for every odd number $i \in \{1, 2, \ldots, 2m\}$, $\mu_i(G)$ and $\mu_{i+1}(G)$ are distinct, key-equal atoms.*

Proof. We show the first item; the proofs of the other items are symmetrical. Let i be an even number in $\{1, 2, \ldots, 2m\}$. Since $\mathsf{KVars}(F) \subseteq F^{+,q}$ is obvious, our construction guarantees that $\theta_i(F)$ and $\theta_{i+1}(F)$ are key-equal. We show next that $\theta_i(F)$ and $\theta_{i+1}(F)$ are distinct.

Let L be the first label on the path from F to G. From $F \overset{\rho}{\rightsquigarrow} G$, it follows $L \nsubseteq F^{+,q}$. We can assume a variable $u \in L \setminus F^{+,q}$. Since $u \in L$, we have $u \in \mathsf{Vars}(F)$. Since $u \notin F^{+,q}$, our construction ensures $\theta_i(u) \neq \theta_{i+1}(u)$. Consequently, $\theta_i(F) \neq \theta_{i+1}(F)$. □

Property 5.3. *For every pair of distinct, key-equal atoms $A, B \in \mathbf{db}$, exactly one of the following holds.*

(1) *for some even number $i \in \{1, 2, \ldots, 2m\}$, $\{A, B\} = \{\theta_i(F), \theta_{i+1}(F)\}$;*
(2) *for some even number $i \in \{1, 2, \ldots, 2m\}$, $\{A, B\} = \{\mu_i(F), \mu_{i+1}(F)\}$;*
(3) *for some odd number $i \in \{1, 2, \ldots, 2m\}$, $\{A, B\} = \{\theta_i(G), \theta_{i+1}(G)\}$; or*
(4) *for some odd number $i \in \{1, 2, \ldots, 2m\}$, $\{A, B\} = \{\mu_i(G), \mu_{i+1}(G)\}$.*

Proof. Assume $A, B \in \mathbf{db}$ are distinct, key-equal atoms having the same relation name as some atom H of q. By construction (see Figure 4), $\mathsf{KVars}(H) \subseteq F^{+,q}$ or $\mathsf{KVars}(H) \subseteq G^{+,q}$ (or both). Assume $\mathsf{KVars}(H) \subseteq F^{+,q}$ (the case where $\mathsf{KVars}(H) \subseteq G^{+,q}$ is symmetrical).

We show that $H = F$ or $H = G$. Assume, towards a contradiction, $H \neq F$ and $H \neq G$. Since $\mathcal{K}(q \setminus \{F\})$ contains the functional dependency $\mathsf{KVars}(H) \rightarrow \mathsf{Vars}(H)$, we have $\mathsf{Vars}(H) \subseteq F^{+,q}$. By our construction, the only possible way of having two distinct key-equal atoms is then $\mathsf{KVars}(H) \subseteq \mathsf{vars}(\vec{y})$ and $\mathsf{Vars}(H) \cap \mathsf{vars}(\vec{x}) \neq \{\}$. From $\mathsf{KVars}(H) \subseteq \mathsf{vars}(\vec{y})$, it follows $\mathsf{KVars}(H) \subseteq G^{+,q}$. Since $\mathcal{K}(q \setminus \{G\})$ contains the functional dependency $\mathsf{KVars}(H) \rightarrow \mathsf{Vars}(H)$, we have $\mathsf{Vars}(H) \subseteq G^{+,q}$. Since $\mathsf{Vars}(H) \subseteq F^{+,q}$, it follows $\mathsf{Vars}(H) \subseteq \mathsf{vars}(\vec{y})$, hence $\mathsf{Vars}(H) \cap \mathsf{vars}(\vec{x}) = \{\}$, a contradiction. We conclude by contradiction that $H = F$ or $H = G$.

Assume A, B have the same relation name as F (the case where A, B have the same relation name as G is symmetrical). Since $G \overset{\rho}{\rightsquigarrow} F$, we have $\mathsf{KVars}(F) \nsubseteq G^{+,q}$ by item (1) in Lemma 4.9. Hence, we can assume a variable $x \in \mathsf{KVars}(F) \cap \mathsf{vars}(\vec{x})$. From our construction, it follows that for some even $i \in \{1, 2, \ldots, 2m\}$, we have either $\{A, B\} = \{\theta_i(F), \theta_{i+1}(F)\}$ or $\{A, B\} = \{\mu_i(F), \mu_{i+1}(F)\}$.

By a symmetrical reasoning, if A, B are key-equal atoms with the same relation name as G, then for some odd $i \in \{1, 2, \ldots, 2m\}$, we have either $\{A, B\} = \{\theta_i(G), \theta_{i+1}(G)\}$ or $\{A, B\} = \{\mu_i(G), \mu_{i+1}(G)\}$. □

The key-equal atoms are schematized in Figure 5.

Property 5.4. *For every valuation ω over U such that $\omega(q) \subseteq \mathbf{db}$, there exists $\zeta \in I$ such that $\omega(F) = \zeta(F)$ and $\omega(G) = \zeta(G)$.*

Proof. Let ω be a valuation over U such that $\omega(q) \subseteq \mathbf{db}$. Assume without loss of generality $i \in \{1, 2, \ldots, 2m + 1\}$ such that $\omega(F) = \theta_i(F)$ (the case where $\omega(F) = \mu_i(F)$ is symmetrical). Assume that for some edge $H \overset{L}{\frown} J$ on the path between F and G in ρ, we have $\omega(H) = \theta_i(H)$ and $\omega(J) = \zeta(J)$ for some $\zeta \in I$. Since $F \overset{\rho}{\rightsquigarrow} G$ and $G \overset{\rho}{\rightsquigarrow} F$, we have $L \nsubseteq F^{+,q}$ and $L \nsubseteq G^{+,q}$. Two cases can occur.

(1) $L \nsubseteq F^{+,q} \cup G^{+,q}$. We can assume $u \in L$ such that $u \in \mathsf{vars}(\vec{u})$. Clearly, $u \in \mathsf{Vars}(H)$ and $u \in \mathsf{Vars}(J)$. From $\omega(H) = \theta_i(H)$, it follows $\omega(u) = \theta_i(u)$. From $\omega(J) = \zeta(J)$, it follows $\omega(u) = \zeta(u)$. Consequently, $\theta_i(u) = \zeta(u)$. Since no two distinct tuples of I agree on u, we have $\theta_i = \zeta$.
(2) $L \subseteq F^{+,q} \cup G^{+,q}$. In this case, we can assume $x, z \in L$ such that $z \notin F^{+,q}$ and $x \notin G^{+,q}$. That is, $x \in \mathsf{vars}(\vec{x})$ and $z \in \mathsf{vars}(\vec{z})$. By the same reasoning as before, $\theta_i(z) = \zeta(z)$ and $\theta_i(x) = \zeta(x)$. Since no two distinct tuples of I agree on both x and z, we have $\theta_i = \zeta$.

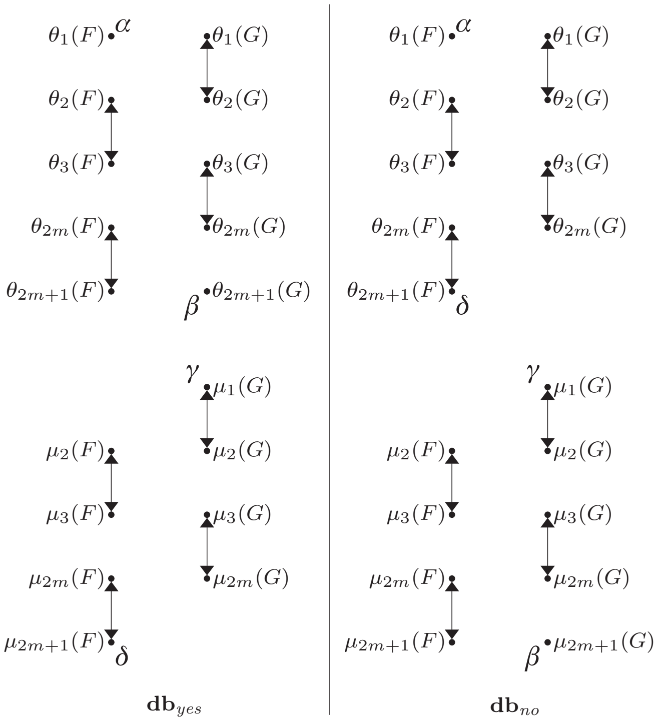


Fig. 6. Case $m = 2$. The figure shows the atoms having the same relation name as F or G in $\mathbf{db}_{yes}$ (left) and in $\mathbf{db}_{no}$ (right). Conflicting, key-equal atoms are linked by a double-arrowed edge.

We conclude $\zeta = \theta_i$. It follows that $\omega(J) = \theta_i(J)$ for every atom J on the path from F to G. In particular, $\omega(G) = \theta_i(G)$. Thus, for every valuation ω over U, $\omega(q) \subseteq \mathbf{db}$ implies that for some $\zeta \in I$, $\omega(F) = \zeta(F)$ and $\omega(G) = \zeta(G)$. □

The databases $\mathbf{db}_{yes}$ and $\mathbf{db}_{no}$ are defined as follows.

$$\begin{aligned}\mathbf{db}_{yes} &= \mathbf{db} \setminus \{\mu_1(F), \mu_{2m+1}(G)\}\\ \mathbf{db}_{no} &= \mathbf{db} \setminus \{\mu_1(F), \theta_{2m+1}(G)\}\end{aligned}$$

The difference between $\mathbf{db}_{yes}$ and $\mathbf{db}_{no}$ is in the atoms having the same relation name as F or G, as schematized in Figure 6 for the case $m = 2$. We have the following.

(1) $\mathbf{db}_{no} \notin$ CERTAINTY(q). Let **rep** be a repair of $\mathbf{db}_{no}$ containing as subsets both $\{\theta_1(F), \theta_2(G), \theta_3(F), \theta_4(G), \ldots, \theta_{2m}(G), \theta_{2m+1}(F)\}$ and $\{\mu_1(G), \mu_2(F), \mu_3(G), \mu_4(F), \ldots, \mu_{2m}(F), \mu_{2m+1}(G)\}$. Intuitively, with respect to $\mathbf{db}_{no}$ in Figure 6, **rep** never selects two atoms that appear on the same horizontal line. It follows from Property 5.3 that such repair **rep** exists. By Property 5.4, $\mathbf{rep} \not\models q$.

(2) $\mathbf{db}_{yes} \in$ CERTAINTY(q). Clearly, $\mathbf{db}_{\theta} \subseteq \mathbf{db}_{yes}$ and every repair of $\mathbf{db}_{yes}$ contains a repair of $\mathbf{db}_{\theta}$. It can be easily seen that for every repair **rep** of $\mathbf{db}_{\theta}$, there exists $i \in \{1, 2, \ldots, 2m + 1\}$ such that $\theta_i(q) \subseteq \mathbf{rep}$. Intuitively, with respect to $\mathbf{db}_{yes}$ in Figure 6, it is impossible to select one atom from every pair of key-equal atoms without ending up with two atoms that appear on the same horizontal line.

Finally, we show that $\mathbf{db}_{yes}$ and $\mathbf{db}_{no}$ are indistinguishable by ψ. Notice that if $R(s_1, \ldots, s_r) \in q$ and s_i is a constant or a variable in $\vec{y}$ $(1 \leq i \leq r)$, then all R-atoms of $\mathbf{db}_{yes}$ agree on position i (likewise for $\mathbf{db}_{no}$). Such positions, called *constant positions*

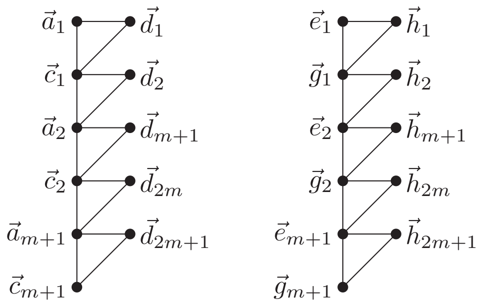


Fig. 7. Case $m = 2$. Schematized Gaifman graph of I in case $\vec{y}$ is empty. An edge between two vectors means that every value in either vector is adjacent with each value in the other vector.

hereafter, cannot be used for distinguishing $\mathbf{db}_{yes}$ and $\mathbf{db}_{no}$. In the following we assume that there are no constant positions (in particular, $\vec{y}$ is empty).

Let $\mathcal{G}(I)$ denote the Gaifman graph of I, which is schematized in Figure 7. For every two constants k, l that occur in I (see Figure 4), we denote by $d_I(k,l)$ the distance between k and l in $\mathcal{G}(I)$. Likewise for the Gaifman graphs of $\mathbf{db}_{yes}$ and $\mathbf{db}_{no}$. Since $\mathcal{G}(\mathbf{db}_{yes})$ and $\mathcal{G}(\mathbf{db}_{no})$ are subgraphs of $\mathcal{G}(I)$, it follows $d_I(k,l) \leq d_{\mathbf{db}_{yes}}(k,l)$ and $d_I(k,l) \leq d_{\mathbf{db}_{no}}(k,l)$.

It is easy to verify that $d_I(\vec{a}_1\vec{c}_1\vec{d}_1, \vec{a}_{m+1}\vec{c}_{m+1}\vec{d}_{2m+1}) = d_I(\vec{c}_1, \vec{a}_{m+1}) = 2m - 1$. Assume constant k such that $d_I(k, \vec{a}_1\vec{c}_1\vec{d}_1) \leq d$, where d is the Hanf-locality rank of ψ. By the triangle inequality, $d_I(k, \vec{a}_{m+1}\vec{c}_{m+1}\vec{d}_{2m+1}) \geq 2m-1-d$. From $m > d$ (see the choice of m at the beginning of the current proof), it follows $2m - 1 > 2d$, hence $d_I(k, \vec{a}_{m+1}\vec{c}_{m+1}\vec{d}_{2m+1}) > d$.

Call two constants k and l *homologous* if k and l occur in the same column of I. It can now be easily verified that each isomorphism type τ of d-neighborhood has the same number of constants realizing τ in $\mathbf{db}_{yes}$ and $\mathbf{db}_{no}$ (intuitively, in Figure 6, Greek letters denote positions in $\mathbf{db}_{yes}$ and $\mathbf{db}_{no}$ that are indistinguishable by ψ; the important thing to notice is that β and δ are swapped).

—*Constants close to α*. Every constant k such that $d_I(k, \vec{a}_1\vec{c}_1\vec{d}_1) \leq d$ realizes the same isomorphism type of d-neighborhood in $\mathbf{db}_{yes}$ and $\mathbf{db}_{no}$.
—*Constants close to β*. Let k and l be homologous constants such that $d_I(k, \vec{a}_{m+1}\vec{c}_{m+1}\vec{d}_{2m+1}) = d_I(l, \vec{e}_{m+1}\vec{g}_{m+1}\vec{h}_{2m+1}) \leq d$. Then, the d-neighborhood of k in $\mathbf{db}_{yes}$ is isomorphic to the d-neighborhood of l in $\mathbf{db}_{no}$.
—*Constants close to γ*. Every constant k such that $d_I(k, \vec{e}_1\vec{g}_1\vec{h}_1) \leq d$ realizes the same isomorphism type of d-neighborhood in $\mathbf{db}_{yes}$ and $\mathbf{db}_{no}$.
—*Constants close to δ*. Let k and l be homologous constants such that $d_I(k, \vec{e}_{m+1}\vec{g}_{m+1}\vec{h}_{2m+1}) = d_I(l, \vec{a}_{m+1}\vec{c}_{m+1}\vec{d}_{2m+1}) \leq d$. Then, the d-neighborhood of k in $\mathbf{db}_{yes}$ is isomorphic to the d-neighborhood of l in $\mathbf{db}_{no}$.

Finally, two homologous constants not referred to in the preceding items (i.e., constants far removed from the positions α, β, γ, δ) realize the same isomorphism type of d-neighborhood in $\mathbf{db}_{yes}$ and $\mathbf{db}_{no}$. It follows $\mathbf{db}_{yes} \leftrightarrows_d \mathbf{db}_{no}$. Since d is the Hanf-locality rank of ψ, it follows $\mathbf{db}_{yes} \models \psi \iff \mathbf{db}_{no} \models \psi$.

Technically, the absence of constant positions is important in the aforesaid application of Hanf-locality. Indeed, assume that all R-atoms of $\mathbf{db}_{yes}$ agree on position i, that is, $R(s_1, \ldots, s_r), R(t_1, \ldots, t_r) \in \mathbf{db}_{yes}$ implies $s_i = t_i$ $(1 \leq i \leq r)$. Then, for every constant a that occurs in some R-atom of $\mathbf{db}_{yes}$, the 2-neighborhood of a includes all R-atoms of $\mathbf{db}_{yes}$ (likewise for $\mathbf{db}_{no}$). To show that the theorem remains valid in the presence of constant positions, we reason as follows. Let k be the quantifier rank of ψ. Our results so far imply that in the absence of constant positions, the value of m

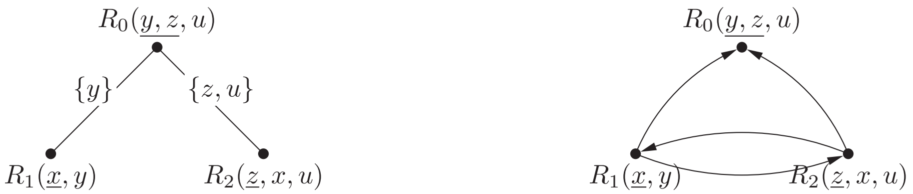


Fig. 8. Intersection tree ρ_8 (left) and attack graph (right) for the cyclic query q_8. The attack graph is cyclic. Notice that ρ_8 is not a join tree.

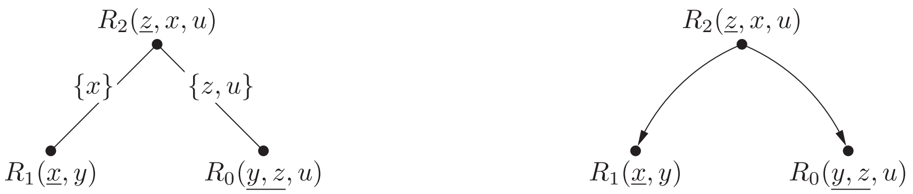


Fig. 9. Another intersection tree ρ'_8 (left) and attack graph (right) for the cyclic query q_8. The attack graph is acyclic. Notice that ρ'_8 is not a join tree.

can be chosen sufficiently large such that the duplicator has a winning strategy in a k-round Ehrenfeucht-Fraïssé game on $\mathbf{db}_{yes}$ and $\mathbf{db}_{no}$. This winning strategy can be easily extended to deal with constant positions: if the spoiler selects a constant that occurs at a constant position in one database, then the duplicator selects the same constant in the other database. This concludes the proof. □

Theorem 5.1 applies to both cyclic and acyclic queries. Consider the cyclic query $q_8 = \{R_0(\underline{y,z},u), R_1(\underline{x},y), R_2(\underline{z},x,u)\}$. Figure 8 shows an intersection tree for q_8 whose attack graph is cyclic. It follows that CERTAINTY(q_8) is not first-order expressible.

A conjunctive query with $n \geq 2$ atoms has n^{n-2} distinct intersection trees [Cayley 1889], and it happens that some, but not all, intersection trees have an attack graph with a cycle of size 2. This is illustrated in Figure 9, which shows that the query q_8 also has an intersection tree with an acyclic attack graph. Nevertheless, the focus in the remainder of this article will be on intersection trees that satisfy the Connectedness Condition, also called join trees. We will show hereafter that if a conjunctive query q has a join tree (and hence is acyclic), then all join trees for q have the same attack graph. What is more, we will show that for acyclic queries, a stronger converse of Theorem 5.1 is true: if an acyclic Boolean conjunctive query q, without self-join, has no certain first-order rewriting, then each join tree for q has an attack graph with a cycle of size 2 (all these attack graphs are actually identical). This will eventually lead to a test for first-order expressibility of CERTAINTY(q) that runs in quadratic time in the length of q.

6. ATTACK GRAPHS OF ACYCLIC CONJUNCTIVE QUERIES

From now on, we focus on conjunctive queries that have a join tree, which are called acyclic in the literature [Beeri et al. 1983]. In general, an acyclic conjunctive query can have different join trees; we show that all these join trees have the same attack graph.

THEOREM 6.1. *Let q be an acyclic Boolean conjunctive query. Let τ_1 and τ_2 be two join trees for q. The attack graphs of τ_1 and τ_2 are identical.*

PROOF. Let F, G be distinct atoms of q such that $F \overset{\tau_1}{\rightsquigarrow} G$. We show $F \overset{\tau_2}{\rightsquigarrow} G$. The proof runs by induction on the size of $[F, G]_{\tau_1}$.

For the induction basis, let $[F, G]_{\tau_1} = \{F, G\}$. Thus, the join tree τ_1 contains an edge between F and G. Since $F \overset{\tau_1}{\rightsquigarrow} G$, we can assume a variable $x \in \mathsf{Vars}(F) \cap \mathsf{Vars}(G)$ such that $x \notin F^{+,q}$. By the *Connectedness Condition*, the variable x is an element of every label on the unique path between F and G in τ_2. It follows $F \overset{\tau_2}{\rightsquigarrow} G$.

For the induction step, assume that the size of $[F, G]_{\tau_1}$ is k with $k > 2$. Let $H \in [F, G]_{\tau_1}$ such that τ_1 contains an edge between H and G. That is, H is the atom on the path between F and G in τ_1 such that H is incident with G. Notice the following.

—Since $F \overset{\tau_1}{\rightsquigarrow} G$, we can assume a variable $x \in \mathsf{Vars}(H) \cap \mathsf{Vars}(G)$ such that $x \notin F^{+,q}$. By the *Connectedness Condition*, the variable x is an element of every label on the unique path between H and G in τ_2.

—By item (2) in Lemma 4.9, $F \overset{\tau_1}{\rightsquigarrow} H$. By the induction hypothesis, $F \overset{\tau_2}{\rightsquigarrow} H$. Thus, every label on the unique path between F and H in τ_2 contains a variable that does not belong to $F^{+,q}$.

Consequently, every label on the unique path between F and G in τ_2 contains a variable that does not belong to $F^{+,q}$. It follows $F \overset{\tau_2}{\rightsquigarrow} G$.

By symmetry, for all distinct atoms F, G of q, $F \overset{\tau_2}{\rightsquigarrow} G$ implies $F \overset{\tau_1}{\rightsquigarrow} G$. Thus, the attack graphs of τ_1 and τ_2 contain the same directed edges. This concludes the proof. □

Theorem 6.1 has the important consequence that we can unambiguously talk about the attack graph of an acyclic Boolean conjunctive query, in the sense of the following definition.

Definition 6.2. Let q be an acyclic Boolean conjunctive query. The *attack graph of* q is defined as the attack graph of any join tree τ for q.

We will write $F \overset{q}{\rightsquigarrow} G$ if the attack graph of q contains a directed edge from F to G. If q is clear from the context, we write $F \rightsquigarrow G$ instead of $F \overset{q}{\rightsquigarrow} G$.

We now provide a strong intuition underlying attack graphs, by showing an equivalence between edges in the attack graph and the existence of particular databases, called witness databases.

Definition 6.3. Let q be an acyclic Boolean conjunctive query, without self-join, containing distinct atoms $F = R(\underline{\vec{x}}, \vec{y})$ and $G = S(\underline{\vec{u}}, \vec{w})$. Let U be the set of variables that occur in q. A *witness database for* (F, G) is a database $\mathbf{db}$ with two distinct, key-equal R-atoms (call them A_1 and A_2) and two S-atoms (call them B_1 and B_2) that are not key-equal such that:

(1) $\mathbf{db}$ has exactly two repairs, namely $\mathbf{db} \setminus \{A_1\}$ and $\mathbf{db} \setminus \{A_2\}$, both satisfying q;
(2) there exists a valuation θ_1 over U such that $A_1, B_1 \in \theta_1(q) \subseteq \mathbf{db}$;
(3) there exists a valuation θ_2 over U such that $A_2, B_2 \in \theta_2(q) \subseteq \mathbf{db}$; and
(4) there exists no valuation μ over U such that $A_1, B_2 \in \mu(q) \subseteq \mathbf{db}$ or $A_2, B_1 \in \mu(q) \subseteq \mathbf{db}$.

The last three items in Definition 6.3 say that A_1 "joins" with B_1 but not with B_2; and that A_2 "joins" with B_2 but not with B_1. For example, given the query q_6 of Figure 2, a witness database for $\big(R_0(\underline{x}, y), R_1(\underline{y}, x)\big)$ is $\{R_0(\underline{a}, 1), R_0(\underline{a}, 2), R_1(\underline{1}, a), R_1(\underline{2}, a), R_2(\underline{a}, a)\}$.

PROPOSITION 6.4. *Let q be an acyclic Boolean conjunctive query, without self-join. Let F, G be two distinct atoms of q. The following two statements are equivalent.*

(*1*) *F attacks G.*
(*2*) *There exists a witness database for (F, G).*

PROOF. Let U be the set of variables that occur in q.

$\boxed{1 \Rightarrow 2}$ Let θ_1, θ_2 be valuations over U such that for every $x \in U$, $\theta_1(x) = \theta_2(x)$ if and only if $x \in F^{+,q}$. Let $\mathbf{db} = \theta_1(q) \cup \theta_2(q)$. Let $A_1 = \theta_1(F)$, $A_2 = \theta_2(F)$, $B_1 = \theta_1(G)$, and $B_2 = \theta_2(G)$. We prove that $\mathbf{db}$ is a witness database for (F, G), by showing that it satisfies all conditions in Definition 6.3.

(1) Since $\mathsf{KVars}(F) \subseteq F^{+,q}$ is obvious, A_1 and A_2 are key-equal. Since $F \rightsquigarrow G$, $\mathsf{Vars}(F) \not\subseteq F^{+,q}$. It follows $A_1 \neq A_2$. Let $H \in q$ such that $H \neq F$. Two cases can occur.
—If $\mathsf{KVars}(H) \subseteq F^{+,q}$, then $\mathsf{Vars}(H) \subseteq F^{+,q}$, hence $\theta_1(H) = \theta_2(H)$.
—If $\mathsf{KVars}(H) \not\subseteq F^{+,q}$, then $\theta_1(H)$ and $\theta_2(H)$ are not key-equal. In particular, since $F \rightsquigarrow G$, we have $\mathsf{KVars}(G) \not\subseteq F^{+,q}$ by Lemma 4.9. Thus B_1 and B_2 are not key-equal.
Consequently, A_1 and A_2 are the only distinct, key-equal atoms of $\mathbf{db}$. It follows that $\mathbf{db} \setminus \{A_1\}$ and $\mathbf{db} \setminus \{A_2\}$ are the two repairs of $\mathbf{db}$. Both repairs satisfy q, because $\theta_2(q) \subseteq \mathbf{db} \setminus \{A_1\}$ and $\theta_1(q) \subseteq \mathbf{db} \setminus \{A_2\}$.
(2) Obviously, $A_1, B_1 \in \theta_1(q) \subseteq \mathbf{db}$.
(3) Obviously, $A_2, B_2 \in \theta_2(q) \subseteq \mathbf{db}$.
(4) Assume that μ is a valuation such that $A_1 \in \mu(q) \subseteq \mathbf{db}$. We show that $\mu(G) = B_1$. Let τ be a join tree for q. Let $F = H_0 \overset{L_1}{\frown} H_1 \overset{L_2}{\frown} H_2 \ldots \overset{L_n}{\frown} H_n = G$ be the path in τ between F and G. We show by induction on increasing i that for each $i \in \{0, 1, \ldots, n\}$, $\mu(H_i) = \theta_1(H_i)$. The induction basis, $i = 0$, holds obviously. For the induction step, $i \rightarrow i + 1$, the induction hypothesis is $\mu(H_i) = \theta_1(H_i)$. Since $F \overset{\tau}{\rightsquigarrow} G$, we can assume $x \in L_{i+1} = \mathsf{Vars}(H_i) \cap \mathsf{Vars}(H_{i+1})$ such that $x \notin F^{+,q}$, hence $\theta_1(x) \neq \theta_2(x)$. From $\mu(H_i) = \theta_1(H_i)$, it follows $\mu(x) = \theta_1(x)$. If $\mu(H_{i+1}) = \theta_2(H_{i+1})$, then $\mu(x) = \theta_2(x)$, a contradiction. We conclude by contradiction that $\mu(H_{i+1}) = \theta_1(H_{i+1})$, so the induction step holds. For $i = n$, we obtain $\mu(G) = B_1$.
By symmetry, if μ is a valuation such that $A_2 \in \mu(q) \subseteq \mathbf{db}$, then $\mu(G) = B_2$.

So it is correct to conclude that $\mathbf{db}$ is a witness database for (F, G).

$\boxed{2 \Rightarrow 1}$ Let $\mathbf{db}_0$ be a witness database for (F, G). We can assume valuations θ_1 and θ_2 as in Definition 6.3, with $A_1 = \theta_1(F)$, $A_2 = \theta_2(F)$, $B_1 = \theta_1(G)$, and $B_2 = \theta_2(G)$. It is obvious that $\mathbf{db} = \theta_1(q) \cup \theta_2(q)$ is a witness database for (F, G). Let $q' = q \setminus \{F\}$ and $\mathbf{db}' = \mathbf{db} \setminus \{A_1, A_2\}$. Since $\mathbf{db}'$ is consistent by the first item in Definition 6.3 and since $\theta_1(q'), \theta_2(q') \subseteq \mathbf{db}'$, it follows by Lemma 4.3 that $\{\theta_1, \theta_2\} \models \mathcal{K}(q')$.

Assume towards a contradiction $F \not\rightsquigarrow G$. Let τ be a join tree for q. We can assume an edge e with label L on the unique path in τ between F and G such that $L \subseteq F^{+,q}$. Let τ_F and τ_G be the two join trees obtained from τ by removing the edge e, such that $F \in \tau_F$ and $G \in \tau_G$. This corresponds to the situation depicted in Figure 12. Let μ be the valuation over U such that for every $x \in U$,

$$\mu(x) = \begin{cases} \theta_1(x) & \text{if } x \text{ occurs in } \tau_F \\ \theta_2(x) & \text{if } x \text{ occurs in } \tau_G. \end{cases}$$

Notice that if x occurs in both τ_F and τ_G, then, by the *Connectedness Condition*, x occurs in L. Since A_1 and A_2 are key-equal, θ_1 and θ_2 agree on $\mathsf{KVars}(F)$. From $\{\theta_1, \theta_2\} \models \mathcal{K}(q \setminus \{F\})$, it follows that θ_1 and θ_2 agree on $F^{+,q}$. Since $L \subseteq F^{+,q}$, valuations θ_1 and θ_2 agree on each $x \in L$. Thus, μ is well-defined. We have $\mu(F), \mu(G) \in \mu(q) \subseteq \mathbf{db}$ with

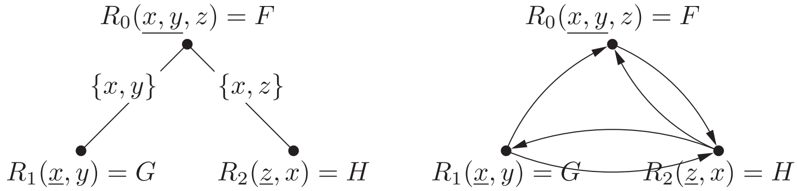

$R_0(\underline{x, y}, z) = F$
$\{x, y\}$
$\{x, z\}$
$R_1(\underline{x}, y) = G$
$R_2(\underline{z}, x) = H$
$R_0(\underline{x, y}, z) = F$
$R_1(\underline{x}, y) = G$
$R_2(\underline{z}, x) = H$

Fig. 10. Join tree τ_9 (left) and attack graph (right) for query q_9. The attack graph is cyclic.

$\mu(F) = A_1$ and $\mu(G) = B_2$, contradicting the fourth item in Definition 6.3. We conclude by contradiction $F \rightsquigarrow G$. □

7. ACYCLIC CONJUNCTIVE QUERIES WITH CYCLIC ATTACK GRAPHS

It is important to understand that the attack graph of an acyclic conjunctive query can be cyclic (see Figure 3) or acyclic (see Figure 2). In this section, we show that if the attack graph of an acyclic conjunctive query has a cycle, then it has a cycle of size 2.

Definition 7.1. Let q be an acyclic Boolean conjunctive query. Let $F_0 \rightsquigarrow F_1 \rightsquigarrow F_2 \ldots \rightsquigarrow F_{n-1} \rightsquigarrow F_0$ be a cycle (of size n) in the attack graph of q. This cycle is said to be *shortest* if the attack graph of q contains no cycle of (strictly) smaller size.

Example 7.2. Consider the join tree τ_9 of query q_9 shown in Figure 10 (left). The attack graph of q_9 is shown at the right. The cycles $G \rightsquigarrow H \rightsquigarrow G$ and $F \rightsquigarrow H \rightsquigarrow F$, both of size 2, are shortest. The cycle $F \rightsquigarrow H \rightsquigarrow G \rightsquigarrow F$ of size 3 is not shortest.

The proof of the following lemma is given in Appendix A.

LEMMA 7.3. *Let q be an acyclic Boolean conjunctive query. Every shortest cycle in the attack graph of q has size* 2.

THEOREM 7.4. *Let q be an acyclic Boolean conjunctive query without self-join. If the attack graph of q is cyclic, then* CERTAINTY(q) *is not first-order expressible.*

PROOF. If the attack graph of q is cyclic, it must contain a shortest cycle of some size n. By Lemma 7.3, $n = 2$. The desired result then follows by Theorem 5.1. □

8. ACYCLIC CONJUNCTIVE QUERIES WITH ACYCLIC ATTACK GRAPHS

We show that the inverse of Theorem 7.4 is also true. That is, if an acyclic conjunctive query q, without self-join, has an acyclic attack graph, then q has a certain first-order rewriting. Moreover, we show how such first-order rewriting can be constructed.

We relax our assumption that all variables in a conjunctive query q are (implicitly) existentially quantified. Hereafter, the notation $q(x_1, x_2, \ldots, x_n)$ is used to indicate that variables $x_1, x_2, \ldots, x_n$ are free in q (while all other variables of q remain existentially quantified). The notion of certain first-order rewriting naturally extends to such nonBoolean queries with free variables.

Definition 8.1. Let $q(x_1, x_2, \ldots, x_n)$ be a conjunctive query with free variables $x_1, x_2, \ldots, x_n$. We say that a first-order formula $\varphi(x_1, x_2, \ldots, x_n)$ is a *certain first-order rewriting* for $q(x_1, x_2, \ldots, x_n)$ if for every database **db**, for all constants $a_1, a_2, \ldots, a_n$,

$$\mathbf{db} \models_{\mathsf{sure}} q(a_1, a_2, \ldots, a_n) \iff \mathbf{db} \models \varphi(a_1, a_2, \ldots, a_n).$$

Definition 8.3 introduces our "rewrite function" Rewrite(F, q) for any Boolean conjunctive query q containing atom F. The definition assumes that we already know a certain first-order rewriting $\varphi(\vec{v})$ for the smaller, nonBoolean query $q \setminus \{F\}$ whose free variables are Vars(F). In front of Theorem 8.13, we will argue that $\varphi(\vec{v})$ can be obtained by recursive application of the same rewrite function. Lemma 8.6 states that, under some condition, our rewrite function produces a certain first-order rewriting for q.

The first-order sentence φ_1 shown in Section 1 was obtained by our rewrite function. More technical details are illustrated by the following example.

Example 8.2. Consider the Boolean singleton query $\{R(\underline{x}, x, y, y, a)\}$ where a is a constant. Obviously, this query is true in every repair of a database **db** if and only if for some constant b, **db** contains an R-atom with primary key value b such that for every R-atom $R(\underline{b}, c_1, c_2, c_3, c_4)$ of **db**, it is the case that $c_1 = b$ (matching variable x) and $c_2 = c_3$ (matching variable y) and $c_4 = a$. This condition is obviously expressed by the following first-order sentence φ.

$$\begin{aligned}\varphi \;=\; &\exists x \exists z_1 \exists z_2 \exists z_3 \exists z_4 \Big(R(\underline{x}, z_1, z_2, z_3, z_4) \\ &\quad \wedge \forall z_1 \forall z_2 \forall z_3 \forall z_4 \Big(R(\underline{x}, z_1, z_2, z_3, z_4) \rightarrow \big(z_1 = x \wedge z_3 = z_2 \wedge z_4 = a \big) \Big) \Big)\end{aligned}$$

Clearly, the beginning of the formula can be simplified to closer reflect the original query.

$$\begin{aligned}\varphi \;\equiv\; &\exists x \exists y \Big(R(\underline{x}, x, y, y, a) \\ &\quad \wedge \forall z_1 \forall z_2 \forall z_3 \forall z_4 \Big(R(\underline{x}, z_1, z_2, z_3, z_4) \rightarrow \big(z_1 = x \wedge z_3 = z_2 \wedge z_4 = a \big) \Big) \Big)\end{aligned}$$

Notice that there is no need for a new variable z_2, because the variable y can be “reused” instead.

$$\begin{aligned}\varphi \;\equiv\; &\exists x \exists y \Big(R(\underline{x}, x, y, y, a) \\ &\quad \wedge \forall z_1 \forall y \forall z_3 \forall z_4 \Big(R(\underline{x}, z_1, y, z_3, z_4) \rightarrow \big(z_1 = x \wedge z_3 = y \wedge z_4 = a \big) \Big) \Big)\end{aligned}$$

The general intuition behind the rewrite function of Definition 8.3 goes as follows. Let $F = R(\underline{\vec{x}}, y_1, y_2, \ldots, y_n)$, an atom of q. It is easy to see that $\mathbf{db} \models_{\mathsf{sure}} q$ if database **db** contains an R-atom $R(\underline{\vec{b}}, \vec{c})$ such that:

—there exists a (unique) valuation θ over $\mathsf{vars}(\vec{x})$ such that $\theta(\vec{x}) = \vec{b}$; and

—for each R-atom $R(\underline{\vec{b}}, c_1, c_2, \ldots, c_n)$ in **db**, we can extend θ to a valuation θ^+ over $\mathsf{Vars}(F)$ such that $\theta^+(y_i) = c_i$ for $1 \leq i \leq n$ and such that $\mathbf{db} \models_{\mathsf{sure}} \theta^+(q')$, where $q' = q \setminus \{F\}$. We assume that R does not occur in q' (no self-join).

Concerning the latter item, to determine whether valuation θ can be extended from $\mathsf{vars}(\vec{x})$ to $\mathsf{Vars}(F)$, we sequentially inspect $\theta^+(y_1)$, $\theta^+(y_2)$, $\theta^+(y_3)$, … in that order. Assume that the value of θ^+ has already been settled for $y_1, y_2, \ldots, y_{i-1}$. Then, two cases are possible for y_i.

(1) If y_i is a variable that does not occur in $\langle \vec{x}, y_1, \ldots, y_{i-1} \rangle$, then we will simply let $\theta^+(y_i)$ be identical to c_i. Correspondingly, in the first item of Definition 8.3, we will let y_i and z_i be identical. In Example 8.2, this happens when y replaces z_2.

(2) Otherwise y_i is a variable that occurs in $\langle \vec{x}, y_1, \ldots, y_{i-1} \rangle$ or y_i is a constant. We distinguish three subcases.

—If $y_i \in \mathsf{vars}(\vec{x})$, then θ maps y_i to a constant. Then θ^+ does not exist unless $c_i = \theta(y_i)$. In Example 8.2, this case leads to the equality $z_1 = x$.

—If y_i is a variable that occurs in $\langle y_1, \dots, y_{i-1}\rangle$ but not in $\vec{x}$, then $\theta^+(y_i)$ has already been settled and it must be the case that $c_i = \theta^+(y_i)$. In Example 8.2, this case leads to the equality $z_3 = y$.

—If y_i is a constant, then it must be the case that $c_i = y_i$, because every valuation is the identity on constants. In Example 8.2, this case leads to the equality $z_4 = a$.

All three subcases are covered by the condition $z_i = y_i$ in the second item of Definition 8.3.

Definition 8.3. Let q be a Boolean conjunctive query without self-join. Let $R(\underline{\vec{x}}, \vec{y})$ be an atom of q, and let $\vec{y} = \langle y_1, y_2, \dots, y_n\rangle$. Notice that $\vec{y}$ can contain constants and repeated variables. Let $\vec{z} = \langle z_1, z_2, \dots, z_n\rangle$ be a sequence of distinct variables and let C be a conjunction of equalities constructed as follows for $1 \leq i \leq n$,

(1) if y_i is a variable that does not occur in the sequence $\langle \vec{x}, y_1, y_2, \dots, y_{i-1}\rangle$, then z_i is identical to y_i;
(2) otherwise z_i is a new variable and C contains $z_i = y_i$.

Let $\vec{v}$ be a sequence of variables that contains exactly once each variable that occurs in $R(\underline{\vec{x}}, \vec{y})$. Let $\varphi(\vec{v})$ be a certain first-order rewriting for $q'(\vec{v})$, where $q'(\vec{v})$ is the nonBoolean conjunctive query whose set of atoms is $q \setminus \{R(\underline{\vec{x}}, \vec{y})\}$ (and whose free variables are $\vec{v}$).[2] Obviously, if q' is empty, then $\varphi = \textbf{true}$. We define

$$\mathsf{Rewrite}(R(\underline{\vec{x}}, \vec{y}), q) \;=\; \exists \vec{v}\Bigg(R(\underline{\vec{x}}, \vec{y}) \wedge \forall \vec{z}\Big(R(\underline{\vec{x}}, \vec{z}) \rightarrow \big(C \wedge \varphi(\vec{v})\big)\Big)\Bigg).$$

If $q'(\vec{v})$ has no certain first-order rewriting, then the value of $\mathsf{Rewrite}(R(\underline{\vec{x}}, \vec{y}), q)$ is undefined.

Example 8.4. Let q be a Boolean conjunctive query without self-join. Let $F = R(\underline{x}, x, y, y, a)$ be an atom of q, where a is a constant. The nonprimary key values in F are $\langle x, y, y, a\rangle$. The sequence $\vec{z}$ of Definition 8.3 becomes $\langle z_1, y, z_3, z_4\rangle$ and C equals $z_1 = x \wedge z_3 = y \wedge z_4 = a$. Notice that there is no variable z_2; the variable y is "reused" instead.

Let $\varphi(x, y)$ be a certain first-order rewriting for $q'(x, y)$, where $q'(x, y)$ is the nonBoolean conjunctive query whose set of atoms is $q \setminus \{F\}$. Then,

$$\mathsf{Rewrite}(F, q) = \exists x \exists y\Bigg(R(\underline{x}, x, y, y, a) \wedge \forall z_1 \forall y \forall z_3 \forall z_4\Big(R(\underline{x}, z_1, y, z_3, z_4) \rightarrow \big(z_1 = x \wedge z_3 = y \wedge z_4 = a \wedge \varphi(x, y)\big)\Big)\Bigg).$$

Our rewrite function $\mathsf{Rewrite}(F, q)$ starts with an existential quantification over the variables in $\mathsf{KVars}(F)$, whose model-theoretic interpretation leads to the following definition.

Definition 8.5. Let q be a Boolean conjunctive query. An atom F of q is said to be *reifiable* if for every database $\mathbf{db}$, $\mathbf{db} \models_{\mathsf{sure}} q$ implies $\mathbf{db} \models_{\mathsf{sure}} \theta(q)$ for some valuation θ over $\mathsf{KVars}(F)$.

LEMMA 8.6. *Let q be a Boolean conjunctive query without self-join. Let F be a reifiable atom of q. If* $\mathsf{Rewrite}(F, q)$ *is defined, then it is a certain first-order rewriting for q.*

[2] In front of Theorem 8.13, it will be argued that the same rewrite function applies to nonBoolean queries by treating free variables as constants.

Proof. Assume $\mathsf{Rewrite}(F, q)$ is defined. Let $F = R(\underline{\vec{x}}, \vec{y})$. Let $\vec{z}$, $\vec{v}$, C, $\varphi(\vec{v})$, and $q'(\vec{v})$ as in Definition 8.3. Let $\mathbf{db}$ be an arbitrary database. We show that $\mathbf{db} \models_{\mathsf{sure}} q$ implies $\mathbf{db} \models \mathsf{Rewrite}(F, q)$ (the converse direction is obvious).

Assume $\mathbf{db} \models_{\mathsf{sure}} q$. Since F is reifiable, we can assume a valuation θ over $\mathsf{vars}(\vec{x})$ such that $\mathbf{db} \models_{\mathsf{sure}} \theta(q)$. Then, for some valuation μ over $\mathsf{vars}(\vec{y}) \setminus \mathsf{vars}(\vec{x})$, $R(\underline{\theta(\vec{x})}, \mu \circ \theta(\vec{y})) \in \mathbf{db}$ and for every valuation ζ over $\mathsf{vars}(\vec{z})$, if $R(\underline{\theta(\vec{x})}, \zeta(\vec{z})) \in \mathbf{db}$, then

$$\begin{array}{l}\text{there exists a valuation } \omega \text{ over } \mathsf{vars}(\vec{y}) \setminus \mathsf{vars}(\vec{x}) \text{ such that } \omega \circ \theta(\vec{y}) = \zeta(\vec{z}) \\ \text{and } \mathbf{db} \models_{\mathsf{sure}} \omega \circ \theta(q').\end{array} \tag{1}$$

Notice that, since $\vec{z}$ is a sequence of distinct variables, every atom $R(\underline{\theta(\vec{x})}, \vec{b})$ can be written as $R(\underline{\theta(\vec{x})}, \zeta(\vec{z}))$ with $\zeta(\vec{z}) = \vec{b}$. Notice also that the foregoing reasoning relies on the fact that relation name R does not occur in $q'(\vec{v})$ (no self-join). Under these definitions, we show the following property.

Property 8.7. *Statement (1) implies both $\zeta \circ \theta(C)$ and $\mathbf{db} \models_{\mathsf{sure}} \zeta \circ \theta(q')$.*

Proof. By Definition 8.3, $\mathsf{vars}(\vec{v}) = \mathsf{vars}(\vec{x}) \cup \mathsf{vars}(\vec{y})$ and $\mathsf{vars}(\vec{y}) \subseteq \mathsf{vars}(\vec{x}) \cup \mathsf{vars}(\vec{z})$. Since $\mathsf{vars}(\vec{v}) \subseteq \mathsf{vars}(\vec{x}) \cup \mathsf{vars}(\vec{z})$, all free variables of q' are replaced by constants in $\zeta \circ \theta(q')$. Furthermore, $\zeta \circ \theta(C)$ is variable-free, because every variable that occurs in C belongs to $\mathsf{vars}(\vec{x}) \cup \mathsf{vars}(\vec{z})$. Assume statement (1).

We first show that $\zeta \circ \theta(C)$ holds. Assume C contains $z_i = y_i$ for some $i \geq 1$. Then, by Definition 8.3, either y_i is a constant or y_i is a variable that occurs in $\langle \vec{x}, y_1, y_2, \ldots, y_{i-1} \rangle$. We need to show

$$\zeta(z_i) = \zeta \circ \theta(y_i). \tag{2}$$

Since $\omega \circ \theta(\vec{y}) = \zeta(\vec{z})$ by (1), it follows

$$\omega \circ \theta(y_i) = \zeta(z_i). \tag{3}$$

Three cases can occur.

—Case y_i is a constant. Then $\omega \circ \theta(y_i) = y_i$. By (3), $\zeta(z_i) = y_i$, which implies (2).
—Case y_i is a variable that occurs in $\vec{x}$. Then, $\omega \circ \theta(y_i) = \theta(y_i)$. By (3), $\zeta(z_i) = \theta(y_i)$, which implies (2).
—Case y_i is a variable that occurs in $\langle y_1, y_2, \ldots, y_{i-1} \rangle$ but not in $\vec{x}$. In this case $\theta(y_i) = y_i$. We can assume an integer $j \in \{1, 2, \ldots, i-1\}$ such that y_i and y_j are identical, and y_j does not occur in $\langle \vec{x}, y_1, \ldots, y_{j-1} \rangle$. By Definition 8.3, z_j and y_j are identical.
Since $\omega \circ \theta(\vec{y}) = \zeta(\vec{z})$ by (1), it follows $\omega \circ \theta(y_j) = \zeta(z_j)$. Since y_i, y_j, and z_j are identical, $\omega \circ \theta(y_i) = \zeta(y_i)$. By (3), $\zeta(z_i) = \zeta(y_i)$, which implies (2).
For the reasoning in the next paragraph, notice also that from (3) and $\omega \circ \theta(y_i) = \omega(y_i)$, it follows $\omega(y_i) = \zeta(z_i)$, hence $\omega(y_i) = \zeta(y_i)$.

So it is correct to conclude that all equalities in $\zeta \circ \theta(C)$ are true.

We next show $\mathbf{db} \models_{\mathsf{sure}} \zeta \circ \theta(q')$. Since $\mathbf{db} \models_{\mathsf{sure}} \omega \circ \theta(q')$ by (1), it suffices to show that for every $y \in \mathsf{vars}(\vec{y}) \setminus \mathsf{vars}(\vec{x})$, $\omega(y) = \zeta(y)$. Assume $i \geq 1$ such that y_i is a variable that does not occur in $\vec{x}$. We need to show $\omega(y_i) = \zeta(y_i)$. Two cases occur: if y_i occurs in $\langle y_1, y_2, \ldots, y_{i-1} \rangle$, then the desired result follows from the third case given earlier; if y_i does not occur in $\langle y_1, y_2, \ldots, y_{i-1} \rangle$, then z_i and y_i are identical, hence $\omega(y_i) = \zeta(y_i)$ by (3). □

Putting it all together, we have

for some valuation θ over $\mathsf{vars}(\vec{x})$,
for some valuation μ over $\mathsf{vars}(\vec{y}) \setminus \mathsf{vars}(\vec{x})$,
  $R(\underline{\theta(\vec{x})}, \mu \circ \theta(\vec{y})) \in \mathbf{db}$ and

$$\text{for every valuation } \zeta \text{ over } \mathsf{vars}(\vec{z}), \text{ if } R(\underline{\theta(\vec{x})}, \zeta(\vec{z})) \in \mathbf{db},$$
$$\text{then } \zeta \circ \theta(C) \text{ and } \mathbf{db} \models_{\mathsf{sure}} \zeta \circ \theta(q').$$

Since $\varphi(\vec{v})$ is a certain first-order rewriting for $q'(\vec{v})$, we have that $\mathbf{db} \models_{\mathsf{sure}} \zeta \circ \theta(q')$ implies $\mathbf{db} \models \varphi(\zeta \circ \theta(\vec{v}))$. Consequently, $\mathbf{db} \models \mathsf{Rewrite}(F, q)$. □

Since Lemma 8.6 only applies to queries q that contain a reifiable atom F, it is important to recognize reifiable atoms. The construct of attack graph is helpful here: Corollary 8.11 states that an atom F of q is reifiable if F is not attacked.

We first show that if an atom F is not attacked, then for each database $\mathbf{db}$, for all repairs $\mathbf{r}$ and $\mathbf{s}$ of $\mathbf{db}$, there exists a repair $\mathbf{rep}$ of $\mathbf{r} \cup \mathbf{s}$ (and hence of $\mathbf{db}$) such that for every valuation θ over $\mathsf{KVars}(F)$, $\mathbf{rep} \models \theta(q)$ implies $\mathbf{r} \models \theta(q)$ and $\mathbf{s} \models \theta(q)$. The construction of such repair $\mathbf{rep}$ is specified next.

Definition 8.8. Let q be a Boolean conjunctive query without self-join. Let U be the set of variables that occur in q. Let $\mathbf{rep}$ be a repair of some database. For an atom F of q, we define

$$\mathsf{Reify}(q, F, \mathbf{rep}) = \{\theta \mid \theta \text{ is a valuation over } \mathsf{KVars}(F) \text{ and } \mathbf{rep} \models \theta(q)\}.$$

We say that an atom $A \in \mathbf{rep}$ is *relevant* for q in $\mathbf{rep}$ if for some valuation θ over U, $A \in \theta(q) \subseteq \mathbf{rep}$. Let $\mathbf{r}$, $\mathbf{s}$ be two repairs of the same database $\mathbf{db}$. A *uniformization* of $[\mathbf{r}, \mathbf{s}]$ is a maximal sequence

$$[\mathbf{r}, \mathbf{s}] = [\mathbf{r}_0, \mathbf{s}_0], [\mathbf{r}_1, \mathbf{s}_1], \ldots, [\mathbf{r}_n, \mathbf{s}_n]$$

where for each $i \in \{0, 1, \ldots, n-1\}$, there exist atoms $A \in \mathbf{r}_i$ and $B \in \mathbf{s}_i$ such that A and B are key-equal and $A \neq B$, and one of the following conditions is true.

—A is relevant for q in $\mathbf{r}_i$ and $\mathbf{r}_{i+1} = (\mathbf{r}_i \setminus \{A\}) \cup \{B\}$ and $\mathbf{s}_{i+1} = \mathbf{s}_i$; or
—B is relevant for q in $\mathbf{s}_i$ and $\mathbf{r}_{i+1} = \mathbf{r}_i$ and $\mathbf{s}_{i+1} = (\mathbf{s}_i \setminus \{B\}) \cup \{A\}$.

That is, in a uniformization we repeatedly replace a relevant atom in either repair with its key-equal (but distinct) atom in the other repair.

Example 8.9. Let $q_3 = \{R(\underline{x}, y), S(\underline{x}, y)\}$ and

$$\begin{aligned}
\mathbf{r}_0 &= \{R(\underline{a}, b), S(\underline{a}, b), R(\underline{c}, 1), S(\underline{c}, 2), R(\underline{e}, f), S(\underline{e}, f)\},\\
\mathbf{s}_0 &= \{R(\underline{a}, 1), S(\underline{a}, 1), R(\underline{c}, d), S(\underline{c}, d), R(\underline{e}, f), S(\underline{e}, f)\}.
\end{aligned}$$

Since $R(\underline{a}, b)$ is relevant for q_3 in $\mathbf{r}_0$, it is replaced with $R(\underline{a}, 1)$, giving

$$\begin{aligned}
\mathbf{r}_1 &= \{R(\underline{a}, 1), S(\underline{a}, b), R(\underline{c}, 1), S(\underline{c}, 2), R(\underline{e}, f), S(\underline{e}, f)\},\\
\mathbf{s}_1 &= \{R(\underline{a}, 1), S(\underline{a}, 1), R(\underline{c}, d), S(\underline{c}, d), R(\underline{e}, f), S(\underline{e}, f)\}.
\end{aligned}$$

Since $S(\underline{a}, 1)$ is relevant for q_3 in $\mathbf{s}_1$, it is replaced with $S(\underline{a}, b)$, giving

$$\begin{aligned}
\mathbf{r}_2 &= \{R(\underline{a}, 1), S(\underline{a}, b), R(\underline{c}, 1), S(\underline{c}, 2), R(\underline{e}, f), S(\underline{e}, f)\},\\
\mathbf{s}_2 &= \{R(\underline{a}, 1), S(\underline{a}, b), R(\underline{c}, d), S(\underline{c}, d), R(\underline{e}, f), S(\underline{e}, f)\}.
\end{aligned}$$

Since $R(\underline{c}, d)$ is relevant for q_3 in $\mathbf{s}_2$, it is replaced with $R(\underline{c}, 1)$, giving

$$\begin{aligned}
\mathbf{r}_3 &= \{R(\underline{a}, 1), S(\underline{a}, b), R(\underline{c}, 1), S(\underline{c}, 2), R(\underline{e}, f), S(\underline{e}, f)\},\\
\mathbf{s}_3 &= \{R(\underline{a}, 1), S(\underline{a}, b), R(\underline{c}, 1), S(\underline{c}, d), R(\underline{e}, f), S(\underline{e}, f)\}.
\end{aligned}$$

The uniformization terminates, because the relevant atoms $R(\underline{e}, f)$ and $S(\underline{e}, f)$ are the same in $\mathbf{r}_3$ and $\mathbf{s}_3$. It is easy to see that every uniformization must eventually terminate because the cardinality of $\mathbf{r}_i \cup \mathbf{s}_i$ decreases at each step.

The proof of the following lemma appears in Appendix B.

LEMMA 8.10. *Let q be an acyclic Boolean conjunctive query without self-join. Let $F \in q$ such that for every $G \in q \setminus \{F\}$, $G \not\rightsquigarrow F$. Let $\mathbf{r}, \mathbf{s}$ be two repairs of the same database $\mathbf{db}$. Let $[\mathbf{r}_n, \mathbf{s}_n]$ be the last element in a uniformization of $[\mathbf{r}, \mathbf{s}]$. Then,*

(*1*) *$\mathbf{r}_n$ is a repair of $\mathbf{db}$; and*
(*2*) $\mathsf{Reify}(q, F, \mathbf{r}_n) \subseteq \mathsf{Reify}(q, F, \mathbf{r}) \cap \mathsf{Reify}(q, F, \mathbf{s})$.

COROLLARY 8.11. *Let q be an acyclic Boolean conjunctive query without self-join. Let $F \in q$ such that for every $G \in q \setminus \{F\}$, $G \not\rightsquigarrow F$. Then F is reifiable.*

PROOF. Let $\mathbf{db}$ be an arbitrary database. By Lemma 8.10, there exists a repair $\mathbf{rep}$ of $\mathbf{db}$ such that $\mathsf{Reify}(q, F, \mathbf{rep}) \subseteq \bigcap\{\mathsf{Reify}(q, F, \mathbf{rep}') \mid \mathbf{rep}' \text{ is a repair of } \mathbf{db}\}$; the latter intersection is finite, because the number of repairs is finite. We distinguish two cases.

—$\mathsf{Reify}(q, F, \mathbf{rep}) \neq \{\}$. Then, we can assume a valuation θ over $\mathsf{KVars}(F)$ such that for each repair $\mathbf{rep}'$ of $\mathbf{db}$, $\mathbf{rep}' \models \theta(q)$. It follows $\mathbf{db} \models_{\mathsf{sure}} \theta(q)$.
—$\mathsf{Reify}(q, F, \mathbf{rep}) = \{\}$. Then, for every valuation θ over $\mathsf{KVars}(F)$, $\mathbf{rep} \not\models \theta(q)$. It follows $\mathbf{rep} \not\models q$. Since $\mathbf{rep}$ is a repair of $\mathbf{db}$, $\mathbf{db} \not\models_{\mathsf{sure}} q$.

Since $\mathbf{db}$ is arbitrary, it follows that F is reifiable in q. □

The full proof of Theorem 8.13 appears in Appendix C. It relies on two main arguments. First, since every directed acyclic graph contains a vertex without an incoming edge, it follows that an acyclic attack graph contains a nonattacked atom, which is reifiable by Corollary 8.11 and thus allows the application of Lemma 8.6. Second, constructing certain first-order rewritings for nonBoolean conjunctive queries $q(x_1, x_2, \ldots, x_n)$ is no more difficult than for Boolean queries (always without self-join). Let $c_1, c_2, \ldots, c_n$ be n new constants. Let q' be the query obtained from q by replacing all occurrences of x_i with c_i $(1 \leq i \leq n)$. Assume we know how to construct a certain first-order rewriting φ for the Boolean conjunctive query q'. A certain first-order rewriting for $q(x_1, x_2, \ldots, x_n)$ can then be obtained from φ by replacing all occurrences of c_i with x_i $(1 \leq i \leq n)$. This is obviously the case because our rewrite function treats constants as generic.

Example 8.12. For $q_3 = \{R(\underline{x}, y), S(\underline{x}, y)\}$, the atom $R(\underline{x}, y)$ is not attacked. A certain first-order rewriting for q_3 is shown next.

$$\begin{aligned}\mathsf{Rewrite}(R(\underline{x}, y), q_3) \;=\; & \exists x \exists y \Big(R(\underline{x}, y) \\ & \wedge \forall y \big(R(\underline{x}, y) \rightarrow \varphi(x, y) \big) \Big)\end{aligned}$$

Here, $\varphi(x, y)$ is a certain first-order rewriting for the query $q'(x, y) = \{S(x, y)\}$, in which x and y are free variables. The formula $\varphi(x, y)$ is obtained by treating x and y as constants in our rewrite function.

$$\varphi(x, y) = S(\underline{x}, y) \wedge \forall z \big(S(\underline{x}, z) \rightarrow z = y \big)$$

It can be verified that $\mathsf{Rewrite}(R(\underline{x}, y), q_3)$ is satisfied by some database $\mathbf{db}$ if and only if for some constants a and b, the following three conditions are satisfied: the database $\mathbf{db}$ contains $R(\underline{a}, b)$ and $S(\underline{a}, b)$; $\mathbf{db}$ contains no R-atom that is distinct but key-equal to $R(\underline{a}, b)$; and $\mathbf{db}$ contains no S-atom that is distinct but key-equal to $S(\underline{a}, b)$.

THEOREM 8.13. *Let q be an acyclic Boolean conjunctive query without self-join. If the attack graph of q contains no cycle, then* $\mathsf{CERTAINTY}(q)$ *is first-order expressible.*

Finally, by combining Theorems 7.4 and 8.13, we obtain the following result.

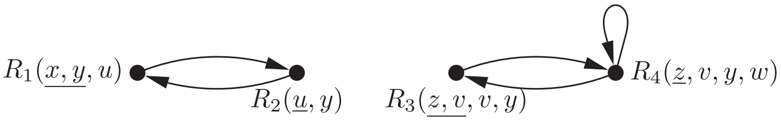

$R_1(\underline{x, y}, u)$
$R_2(\underline{u}, y)$
$R_3(\underline{z, v}, v, y)$
$R_4(\underline{z}, v, y, w)$

Fig. 11. Witness graph of $R_1(\underline{x, y}, u)$.

COROLLARY 8.14. *Let q be an acyclic Boolean conjunctive query without self-join. Then the following two statements are equivalent.*

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

9. THE COMPLEXITY OF DECIDING FIRST-ORDER EXPRESSIBILITY

In this section, we show the following result.

THEOREM 9.1. *Let q be an acyclic Boolean conjunctive query without self-join. First-order expressibility of* CERTAINTY(q) *can be tested in quadratic time in the length of q.*

We first give an alternative characterization of attack graphs and then we provide Algorithm QuadAttack that proves Theorem 9.1.

9.1. Witness Graphs

We show a theoretical result that explains how attack graphs can be built without constructing join trees. The following definition is relative to an acyclic Boolean conjunctive query q without self-join. We also assume a strict linear order, denoted $\prec$, on the atoms of q. In Algorithm QuadAttack, atoms will be represented by integers $1, \ldots, n$.

Definition 9.2. Let U be the set of variables that occur in query q. For every $u \in U$, we define $\mathsf{cycle}(u) = \{(G_1, G_2), (G_2, G_3), \ldots, (G_{\ell-1}, G_\ell), (G_\ell, G_1)\}$ where $G_1 \prec G_2 \prec \cdots \prec G_\ell$ are all the atoms of q in which u occurs, that is, $\{G \in q \mid u \in \mathsf{Vars}(G)\} = \{G_1, \ldots, G_\ell\}$. From a graph perspective, $\mathsf{cycle}(u)$ is a directed cycle graph through all atoms of q that contain u.

For $F \in q$, the *witness graph* of F, denoted $\mathsf{Witness}(F)$, is the following directed graph:

—the vertices of $\mathsf{Witness}(F)$ are the atoms of q; and
—the set of edges of $\mathsf{Witness}(F)$ is $\bigcup\{\mathsf{cycle}(u) \mid u \in U \setminus F^{+,q}\}$.

Example 9.3. It can be verified that $q_{10} = \{R_1(\underline{x, y}, u), R_2(\underline{u}, y), R_3(\underline{z, v}, v, y), R_4(\underline{z}, v, y, w)\}$ is acyclic. Let $F = R_1(\underline{x, y}, u)$ and $U = \{u, v, w, x, y, z\}$, the variables in q_{10}. We have $U \setminus F^{+,q_{10}} = \{u, v, w, z\}$. For each variable in $\{u, v, w, z\}$, the witness graph of F contains a cycle traversing the atoms that contain that variable. Since v and z occur in exactly the same atoms, their cycles coincide. The witness graph of F is shown in Figure 11.

Since distinct edges in the witness graph of F can be associated to distinct occurrences of variables in q, the size of F's witness graph is linearly bounded in the length of q. For example, the self-loop in Figure 11 is associated to the occurrence of the variable w in $R_4(\underline{z}, v, y, w)$. The following theorem establishes an equivalence between attacks by F and reachability by F in the witness graph of F.

THEOREM 9.4. *Let q be an acyclic Boolean conjunctive query without self-join, equipped with a linear order on its atoms. For all $F, G \in q$ with $F \neq G$, the following two conditions are equivalent.*

(*1*) *F attacks G.*
(*2*) *There is a directed path from F to G in* $\mathsf{Witness}(F)$*.*

PROOF. Let U be the set of variables that occur in q. Let τ be a join tree for q.

$\boxed{1 \Rightarrow 2}$ Assume $F \rightsquigarrow G$. Let $H \overset{L}{\frown} J$ be an edge on the unique path in τ that links F and G. We can assume $x \in L$ such that $x \in U \setminus F^{+,q}$. Since $x \in \mathsf{Vars}(H) \cap \mathsf{Vars}(J)$, the graph $\mathsf{Witness}(F)$ contains a cycle that contains H and J. Thus, for every edge $H \overset{L}{\frown} J$ on the path in τ between F and G, there exists a directed path from H to J in $\mathsf{Witness}(F)$. It follows that there is a directed path from F to G in $\mathsf{Witness}(F)$.

$\boxed{2 \Rightarrow 1}$ Assume there is a directed path $(G_0, G_1, \ldots, G_n)$ in $\mathsf{Witness}(F)$ with $G_0 = F$ and $G_n = G$. We can assume without loss of generality that for $0 \leq i < j \leq n$, $G_i \neq G_j$. Let $i \in \{1, \ldots, n\}$. Since (G_{i-1}, G_i) is a directed edge in $\mathsf{Witness}(F)$, we can assume a variable $x_i \in U \setminus F^{+,q}$ such that $(G_{i-1}, G_i) \in \mathsf{cycle}(x_i)$. Since $x_i \in \mathsf{Vars}(G_{i-1}) \cap \mathsf{Vars}(G_i)$, by the *Connectedness Condition* for join trees, x_i occurs in every atom on the unique path in τ that links G_{i-1} and G_i. Since i is an arbitrary element of $\{1, \ldots, n\}$, it can now be easily shown by induction on increasing j that $F \rightsquigarrow G_j$ for all $j \in \{1, \ldots, n\}$. It follows $F \rightsquigarrow G$. □

9.2. Quadratic-Time Algorithm

By Corollary 8.14 and Lemma 7.3, given an acyclic conjunctive query q without self-join, $\mathsf{CERTAINTY}(q)$ is first-order expressible if and only if q's attack graph contains no cycle of size 2. Algorithm QuadAttack builds the attack graph of q and tests the existence of a cycle of size 2. It does so without computing a join tree for q (but we know that such join tree exists if q is acyclic). We will argue that Algorithm QuadAttack runs in quadratic time in the length of q.

The length of q is the number of symbols used to represent q. Algorithm QuadAttack uses the following representation: variables are represented by integers $1, \ldots, m$, and atoms by integers $1, \ldots, n$; KEY and NONKEY are arrays of lists that store for each atom F the variables of $\mathsf{KVars}(F)$ and $\mathsf{Vars}(F) \setminus \mathsf{KVars}(F)$ respectively. Hereinafter, the length of this encoding will be denoted by $\mathsf{len}(q)$. Notice that $\mathsf{len}(q)$ is a lower bound for the size of more detailed representations that keep track of constants or duplicate occurrences of the same variable. Also, converting any other representation into the algorithm's representation should not take more than quadratic time in the length of the original encoding.

Example 9.5. For $q_{10} = \{R_1(\underline{x, y}, u), R_2(\underline{u}, y), R_3(\underline{z, v}, v, y), R_4(\underline{z}, v, y, w)\}$, assume that each R_i-atom is represented by i $(1 \leq i \leq 4)$. Assume that variables u, v, w, x, y, z are represented by $1, 2, 3, 4, 5, 6$ respectively. Then, Algorithm QuadAttack uses the following representation for q (for readability, references to atoms are typeset in bold).

$$\begin{array}{ll}
\mathsf{KEY}[\mathbf{1}] = (4, 5) & \mathsf{NONKEY}[\mathbf{1}] = (1) \\
\mathsf{KEY}[\mathbf{2}] = (1) & \mathsf{NONKEY}[\mathbf{2}] = (5) \\
\mathsf{KEY}[\mathbf{3}] = (2, 6) & \mathsf{NONKEY}[\mathbf{3}] = (5) \\
\mathsf{KEY}[\mathbf{4}] = (6) & \mathsf{NONKEY}[\mathbf{4}] = (2, 3, 5)
\end{array}$$

Note incidentally that there is no need to know that v also occurs at a nonprimary key position in the third atom.

ALGORITHM QuadAttack: Build attack graph and test for its acyclicity

Input: acyclic Boolean conjunctive query q without self-join
Result: the attack graph of q is computed in the adjacency matrix ATTACK. The Boolean variable FO indicates whether CERTAINTY(q) is first-order expressible.
Data:
The following data structures encode all information of q needed in the computation.

—Atoms are encoded by integers $1, 2, \dots, n$.

—Variables are encoded by integers $1, 2, \dots, m$.

—KEY[$1 : n$] is an array of lists of variables such that for $1 \leq F \leq n$, entry KEY[F] contains all variables that occur in the primary key of atom F.

—NONKEY[$1 : n$] is an array of lists of variables such that for $1 \leq F \leq n$, entry NONKEY[F] contains all variables that occur in atom F but do not occur in the primary key of F.

The following data structures will be filled in by the program.

—KEYCOUNT[$1 : n$] is an array of integers such that for $1 \leq F \leq n$, entry KEYCOUNT[F] is the number of distinct variables in the primary key of atom F. KEYCOUNT remains unchanged after initialization.

—COUNT[$1 : n$] is an array of integers.

—KEYCO[$1 : m$] is an array of lists of atoms such that for $1 \leq u \leq m$, entry KEYCO[u] contains all atoms in which variable u occurs at some primary key position. KEYCO remains unchanged after initialization.

—CO[$1 : m$] is an array of lists of atoms such that for $1 \leq u \leq m$, entry CO[u] contains the atoms in which variable u occurs. CO remains unchanged after initialization.

—WITNESSF[$1 : n$] is an *adjacency list* representation of a directed graph whose vertices are the atoms of q. For $1 \leq G \leq n$, entry WITNESSF[G] holds the atoms to which G has an outgoing edge. For each atom F of q, this data structure is reused to store the witness graph of F.

—ATTACK[$1 : n$, $1 : n$] is an *adjacency matrix* representing a directed graph whose vertices are the atoms of q. Entry ATTACK[F, G] is 1 if there is a directed edge from F to G, and is 0 otherwise. This data structure is used to store the attack graph of q.

```
begin
    Initialization;
    BuildAttackGraph;
end
```

The Initialization procedure fills in the following two data structures, obviously in $\mathcal{O}(\mathsf{len}(q))$ time: an array CO of lists that stores for each variable u the atoms in which u occurs; an array KEYCO of lists that stores for each variable u the atoms in which u occurs at some primary key position.

Example 9.6. For the query q_{10} of Example 9.5, these data structures look as follows (see Example 9.5 for the numbering of atoms and variables).

$$\begin{array}{ll}
\mathsf{CO}[1] = (\mathbf{1}, \mathbf{2}) & \mathsf{KEYCO}[1] = (\mathbf{2}) \\
\mathsf{CO}[2] = (\mathbf{3}, \mathbf{4}) & \mathsf{KEYCO}[2] = (\mathbf{3}) \\
\mathsf{CO}[3] = (\mathbf{4}) & \mathsf{KEYCO}[3] = () \\
\mathsf{CO}[4] = (\mathbf{1}) & \mathsf{KEYCO}[4] = (\mathbf{1}) \\
\mathsf{CO}[5] = (\mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{4}) & \mathsf{KEYCO}[5] = (\mathbf{1}) \\
\mathsf{CO}[6] = (\mathbf{3}, \mathbf{4}) & \mathsf{KEYCO}[6] = (\mathbf{3}, \mathbf{4})
\end{array}$$

For every atom F, the Initialization procedure also sets KEYCOUNT[F] to the number of distinct variables in KVars(F). The attack graph will be computed in the variable ATTACK using an adjacency matrix representation [Dasgupta et al. 2008, page 81]. All entries of ATTACK are initialized to 0 in quadratic time in the length of q.

Procedure Initialization

```
foreach $u \leftarrow 1$ to $m$ do
    $\mathsf{KEYCO}[u] \leftarrow \emptyset$;
    $\mathsf{CO}[u] \leftarrow \emptyset$;
end
foreach $F \leftarrow 1$ to $n$ do
    $i \leftarrow 0$;
    foreach $u$ in $\mathsf{KEY}[F]$ do
        add $F$ to $\mathsf{KEYCO}[u]$;
        $i \leftarrow i + 1$;
    end
    $\mathsf{KEYCOUNT}[F] \leftarrow i$;
    foreach $u$ in $\mathsf{KEY}[F] \cup \mathsf{NONKEY}[F]$ do
        add $F$ to $\mathsf{CO}[u]$;
    end
    foreach $G \leftarrow 1$ to $n$ do
        $\mathsf{ATTACK}[F, G] \leftarrow 0$;
    end
end
$\mathsf{FO} \leftarrow$ true;
```

Algorithm QuadAttack exploits Theorem 9.4 in a straightforward manner. After the Initialization procedure, the BuildAttackGraph procedure loops through all atoms of the input query q. For each atom F, the witness graph of F is computed in the variable WITNESSF using an adjacency list representation [Dasgupta et al. 2008, page 82].

Example 9.7. The witness graph of Figure 11 is represented by the following array of adjacency lists (see Example 9.5 for the numbering of atoms).

$$\mathsf{WITNESSF}[\mathbf{1}] = (\mathbf{2})$$
$$\mathsf{WITNESSF}[\mathbf{2}] = (\mathbf{1})$$
$$\mathsf{WITNESSF}[\mathbf{3}] = (\mathbf{4})$$
$$\mathsf{WITNESSF}[\mathbf{4}] = (\mathbf{3}, \mathbf{4})$$

The last entry, for example, indicates that atom **4** has outgoing edges to atoms **3** and **4**.

For every atom G that is distinct but reachable from F in the witness graph of F, a directed edge from F to G is added to the attack graph, and it is checked whether this new edge causes a cycle of size 2 in the attack graph. Whenever such cycle occurs, CERTAINTY(q) is not first-order expressible.

To obtain a quadratic-time algorithm, at most a linear amount of time can be spent per atom in the main loop that constitutes the BuildAttackGraph procedure. Since the witness graph of F has size $\mathcal{O}(\mathsf{len}(q))$, once its adjacency list representation WITNESSF is filled in, the atoms reachable from F can be obtained in linear time using a standard graph traversal algorithm [Dasgupta et al. 2008, page 84]. For each atom reached from F, it takes only constant time to add an edge to the adjacency matrix ATTACK. So it suffices now to show that for each atom F, it takes only linear time to compute the adjacency list representation of F's witness graph.

In order to compute the witness graph of F, we need to know $F^{+,q}$. The computation of $F^{+,q}$ uses a slightly adapted version of the LinClosure algorithm [Beeri and Bernstein 1979, Algorithm 2; Maier 1983, page 66]. The set $F^{+,q}$ is constructed in the variable FPLUS, which is initialized to KEY[F]. Then, for every atom $G \neq F$ with $\mathsf{KVars}(G) = \{\}$, all variables of NONKEY[G] are added to FPLUS. The further computation relies on a counter COUNT[G] for each atom G, which is initialized to the cardinality of $\mathsf{KVars}(G)$.

Procedure BuildAttackGraph

```
foreach $F \leftarrow 1$ to $n$ do
    FPLUS ← KEY[$F$];                  // variable FPLUS is used to compute $F^{+,q}$
    foreach $G \leftarrow 1$ to $n$ do
        WITNESSF[$G$] ← $\emptyset$;
        COUNT[$G$] ← KEYCOUNT[$G$];
        if KEYCOUNT[$G$] $= 0$ and $G \neq F$ then
            FPLUS ← FPLUS ∪ NONKEY[$G$];
        end
    end
    COUNT[$F$] ← $m + 1$;              // high value that will never be decremented to 0
    UPDATE ← FPLUS;
    while UPDATE $\neq \emptyset$ do
        choose a variable $u$ in UPDATE;
        UPDATE ← UPDATE \ $\{u\}$;
        foreach $G$ in KEYCO[$u$] do
            COUNT[$G$] ← COUNT[$G$] $- 1$;
            if COUNT[$G$] $= 0$ then
                ADD ← NONKEY[$G$] \ FPLUS;
                FPLUS ← FPLUS ∪ ADD;
                UPDATE ← UPDATE ∪ ADD;
            end
        end
    end
    foreach $u$ in $\{1, \ldots, m\}$ \ FPLUS do      // construct the witness graph of $F$
        Encode the edges of cycle($u$) in WITNESSF, as follows. If $G_1, G_2, \ldots, G_\ell$ are the atoms in
        CO[$u$] in increasing order, then add $G_1$ to WITNESSF[$G_\ell$], and for $1 \leq i < \ell$, add $G_{i+1}$ to
        WITNESSF[$G_i$].
    end
    foreach atom $G \neq F$ that is reachable from $F$ in the graph represented by WITNESSF do
        ATTACK[$F, G$] ← 1;
        if ATTACK[$G, F$] $= 1$ then
            FO ← false;
        end
    end
end
```

The computation of $F^{+,q}$ loops exactly once through each variable u that has been or will be inserted into FPLUS, and decrements COUNT[G] for every atom G in KEYCO[u]. If COUNT[G] reaches 0 for some $G \neq F$, then we know that all variables of KVars(G) belong to $F^{+,q}$, so we add each variable of NONKEY[G] to FPLUS (if not already present). Rather than testing $G \neq F$ whenever the counter of some G goes to 0, we initialize the counter of F to some high value so that it never gets decremented to 0. Using the same argumentation as for LinClosure [Beeri and Bernstein 1979; Maier 1983], it can be shown that $F^{+,q}$ is correctly computed in the variable FPLUS in $\mathcal{O}(\mathsf{len}(q))$ time.

Once $F^{+,q}$ is known, the witness graph of F is computed in the variable WITNESSF. For each variable u that is not in $F^{+,q}$, we encode in WITNESSF the cycle graph through the atoms in the list CO[u]. Since every edge in every cycle graph can be one-to-one related to some occurrence of some variable in q, filling in WITNESSF takes $\mathcal{O}(\mathsf{len}(q))$ time.

10. CONCLUDING REMARKS

The complexity of the problem CERTAINTY(q) for conjunctive queries q has received a growing research interest in recent years. It is a special case of consistent query

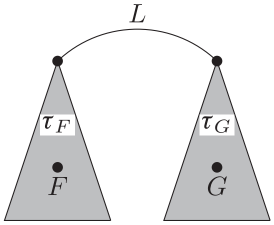


Fig. 12. Join tree τ in the premise of Lemma A.1.

answering, where the only constraints are primary keys. In this article, we provided a syntactic characterization of the frontier between first-order expressible and not first-order expressible cases of $\mathsf{CERTAINTY}(q)$ when q ranges over the class of acyclic conjunctive queries without self-join. This result is of practical interest, because if $\mathsf{CERTAINTY}(q)$ is first-order expressible, then $\mathsf{CERTAINTY}(q)$ can be expressed in SQL and solved using standard database technology. The attack graph turns out to be a useful, novel construct.

Some issues for further research are as follows.

—Can we extend the results to conjunctive queries that are cyclic and/or contain self-joins? Although Theorem 5.1 applies to cyclic queries, it only provides a necessary condition for first-order expressibility of $\mathsf{CERTAINTY}(q)$ when q is cyclic.

—Can we find a syntactic characterization of the frontier between tractable and intractable cases of $\mathsf{CERTAINTY}(q)$ when q ranges over the class of conjunctive queries without self-join? Such characterization is known for queries with exactly two atoms [Kolaitis and Pema 2012] and for the counting variant of $\mathsf{CERTAINTY}(q)$ [Maslowski and Wijsen 2011].

APPENDIXES

A. PROOF OF LEMMA 7.3

We will use two helping lemmas. Lemma A.1 is technical. The situation required by the premise is illustrated in Figure 12: the join tree τ consists of two join trees, τ_F and τ_G, linked by an edge with label L; the atom F occurs in τ_F, and G in τ_G.

LEMMA A.1. *Let q be a Boolean conjunctive query. Let τ be a join tree for q. Let F, G be distinct atoms of q. Let e be an edge with label L on the path between F and G. Let τ_F and τ_G be the join trees obtained from τ by cutting the edge e, such that $F \in \tau_F$ and $G \in \tau_G$. Let U_G be the set of variables that occur in τ_G. For every $q' \subseteq q$, for every $S \subseteq U_G$, if $\mathcal{K}(q') \models \mathsf{KVars}(F) \rightarrow S$, then $\mathcal{K}(q') \models L \rightarrow S$.*

PROOF. Let U be the set of variables that occur in q. The computation of the set $\{x \in U \mid \mathcal{K}(q') \models \mathsf{KVars}(F) \rightarrow x\}$ by means of a standard algorithm [Abiteboul et al. 1995, page 165] corresponds to constructing a maximal sequence (see also the proof of Lemma 4.9)

$$\begin{array}{rcll} \mathsf{KVars}(F) & = & S_0 & H_1 \\ & & S_1 & H_2 \\ & & \vdots & \vdots \\ & & S_{k-1} & H_k \\ & & S_k & \end{array}$$

where

(1) $S_0 \subsetneq S_1 \subsetneq \cdots \subsetneq S_{k-1} \subsetneq S_k$; and

(2) for every $i \in \{1, 2, \dots, k\}$, we have $H_i \in q'$ such that $\mathsf{KVars}(H_i) \subseteq S_{i-1}$ and $S_i = S_{i-1} \cup \mathsf{Vars}(H_i)$.

Then, for every $x \in U$, $\mathcal{K}(q') \models \mathsf{KVars}(F) \rightarrow x$ if and only if $x \in S_k$. We show by induction on increasing i that for every $0 \leq i \leq k$, $\mathcal{K}(q') \models L \rightarrow S_i \cap U_G$.

The induction basis, $i = 0$, holds because $\mathsf{KVars}(F) \cap U_G \subseteq L$ by the *Connectedness Condition* on join trees. For the induction step, $i \rightarrow i + 1$, we distinguish two cases.

(1) H_{i+1} belongs to τ_F. By the *Connectedness Condition*, $\mathsf{Vars}(H_{i+1}) \cap U_G \subseteq L$. Thus, $S_{i+1} \cap U_G \subseteq (S_i \cap U_G) \cup L$. Since $\mathcal{K}(q') \models L \rightarrow S_i \cap U_G$ by the induction hypothesis, $\mathcal{K}(q') \models L \rightarrow (S_i \cap U_G) \cup L$. Consequently, $\mathcal{K}(q') \models L \rightarrow S_{i+1} \cap U_G$.

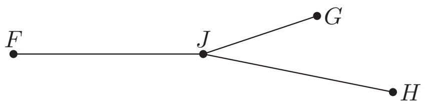
(2) H_{i+1} belongs to τ_G, hence $\mathsf{Vars}(H_{i+1}) \subseteq U_G$. We have $S_{i+1} \cap U_G = (S_i \cap U_G) \cup \mathsf{Vars}(H_{i+1})$. Since $\mathcal{K}(q') \models L \rightarrow S_i \cap U_G$ by the induction hypothesis, it suffices to show $\mathcal{K}(q') \models L \rightarrow \mathsf{Vars}(H_{i+1})$.
Since $\mathsf{KVars}(H_{i+1}) \subseteq S_i \cap U_G$ and $\mathcal{K}(q') \models L \rightarrow S_i \cap U_G$ by the induction hypothesis, we have $\mathcal{K}(q') \models L \rightarrow \mathsf{KVars}(H_{i+1})$. Since $H_{i+1} \in q'$, the set $\mathcal{K}(q')$ contains $\mathsf{KVars}(H_{i+1}) \rightarrow \mathsf{Vars}(H_{i+1})$. Consequently, $\mathcal{K}(q') \models L \rightarrow \mathsf{Vars}(H_{i+1})$.

Consequently, $\mathcal{K}(q') \models L \rightarrow S_k \cap U_G$. This concludes the proof. □

LEMMA A.2. *Let q be an acyclic Boolean conjunctive query. Let F, G, H be distinct atoms of q. If $F \rightsquigarrow G$ and $G \rightsquigarrow H$, then $F \rightsquigarrow H$ or $G \rightsquigarrow F$.*

PROOF. Assume $F \rightsquigarrow G$ and $G \rightsquigarrow H$. Let τ be a join tree for q. We distinguish three cases.

Case $H \in [F, G]_\tau$. By item (2) in Lemma 4.9, $F \overset{\tau}{\rightsquigarrow} H$.
Case $F \in [G, H]_\tau$. By item (2) in Lemma 4.9, $G \overset{\tau}{\rightsquigarrow} F$.
Case $H \notin [F, G]_\tau$ and $F \notin [G, H]_\tau$. Thus, for some $J \notin \{F, H\}$, the join tree τ must contain a subtree of the following form, where straight lines denote paths of some unspecified length. Notice that $J = G$ is possible.



Assume towards a contradiction $F \overset{\tau}{\not\rightsquigarrow} H$ and $G \overset{\tau}{\not\rightsquigarrow} F$.

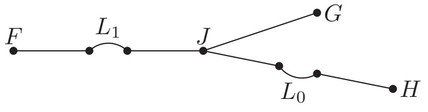
Since $F \overset{\tau}{\rightsquigarrow} G$ and $F \overset{\tau}{\not\rightsquigarrow} H$, we can assume an edge e_0 with label L_0 on the path between J and H such that $\mathcal{K}(q \setminus \{F\}) \models \mathsf{KVars}(F) \rightarrow L_0$. From item (3) in Lemma 4.9, it follows $\mathcal{K}(q \setminus \{F, G\}) \models \mathsf{KVars}(F) \rightarrow L_0$, hence

$$\mathcal{K}(q \setminus \{G\}) \models \mathsf{KVars}(F) \rightarrow L_0. \tag{4}$$

Likewise, since $G \overset{\tau}{\rightsquigarrow} H$ and $G \overset{\tau}{\not\rightsquigarrow} F$, we can assume an edge e_1 with label L_1 on the path between J and F such that

$$\mathcal{K}(q \setminus \{G\}) \models \mathsf{KVars}(G) \rightarrow L_1. \tag{5}$$

The positions of e_0 and e_1 are schematized in the following picture.



From (4) and Lemma A.1, it follows $\mathcal{K}(q \setminus \{G\}) \models L_1 \rightarrow L_0$. Then, by (5) and transitivity, $\mathcal{K}(q \setminus \{G\}) \models \mathsf{KVars}(G) \rightarrow L_0$. But then $G \overset{\tau}{\not\rightsquigarrow} H$, a contradiction. We conclude by contradiction that $F \overset{\tau}{\rightsquigarrow} H$ or $G \overset{\tau}{\rightsquigarrow} F$.

Thus, the desired result holds in all cases. □

The proof of Lemma 7.3 is given next.

Proof of Lemma 7.3. Assume that $F_0 \rightsquigarrow F_1 \rightsquigarrow F_2 \ldots \rightsquigarrow F_{n-1} \rightsquigarrow F_0$ is a shortest cycle of size n in the attack graph of q. We need to show $n = 2$. Assume, towards a contradiction, $n \geq 3$. We distinguish two cases.

Case $F_1 \rightsquigarrow F_0$. Then $F_0 \rightsquigarrow F_1 \rightsquigarrow F_0$ is a cycle of size 2, contradicting that some shortest cycle has size at least 3.

Case $F_1 \not\rightsquigarrow F_0$. By Lemma A.2, $F_0 \rightsquigarrow F_2$. Then $F_0 \rightsquigarrow F_2 \ldots \rightsquigarrow F_{n-1} \rightsquigarrow F_0$ is a cycle of size $n - 1$ in the attack graph of q, again a contradiction.

We conclude $n = 2$ by contradiction. □

B. PROOF OF LEMMA 8.10

We use the following helping lemma.

Lemma B.1. *Let q be an acyclic Boolean conjunctive query without self-join. Let $F \in q$ such that for every $G \in q \setminus \{F\}$, $G \not\rightsquigarrow F$. Let $\mathbf{rep}$ be a repair of some database. Let $A \in \mathbf{rep}$ such that A is relevant for q in $\mathbf{rep}$. Let B be key-equal to A and $\mathbf{rep}_B = (\mathbf{rep} \setminus \{A\}) \cup \{B\}$. Then,* $\mathsf{Reify}(q, F, \mathbf{rep}_B) \subseteq \mathsf{Reify}(q, F, \mathbf{rep})$.

Proof. The proof is obvious if A has the same relation name as F. Assume next that relation names in A and F are different.

Let ζ be a valuation over $\mathsf{KVars}(F)$ such that $\mathbf{rep}_B \models \zeta(q)$. Let U be the set of variables that occur in q. We can assume a valuation ζ^+ over U such that $\zeta^+[\mathsf{KVars}(F)] = \zeta[\mathsf{KVars}(F)]$ and $\zeta^+(q) \subseteq \mathbf{rep}_B$. Thus, ζ^+ extends ζ to U. We need to show $\mathbf{rep} \models \zeta(q)$, which is obvious if $B \notin \zeta^+(q)$. Assume next $B \in \zeta^+(q)$. Since A is relevant for q in $\mathbf{rep}$, we can assume a valuation μ over U such that $A \in \mu(q) \subseteq \mathbf{rep}$. We can assume $G \in q$ such that A, B have the same relation name as G. Thus, $G \neq F$, $A = \mu(G)$, and $B = \zeta^+(G)$. Let $q' = q \setminus \{G\}$. Let $\mathbf{rep}' = \mathbf{rep}_B \setminus \{B\} = \mathbf{rep} \setminus \{A\}$. Since q' contains no atom with the same relation name as G (no self-join), $\zeta^+(q') \subseteq \mathbf{rep}'$ and $\mu(q') \subseteq \mathbf{rep}'$. Moreover, $\zeta^+[\mathsf{KVars}(G)] = \mu[\mathsf{KVars}(G)]$, because A and B are key-equal. Let τ be a join tree for q. Since $G \overset{\tau}{\not\rightsquigarrow} F$, we can assume an edge e with label L on the unique path in τ between G and F such that $L \subseteq G^{+,q}$. Since $\mathcal{K}(q \setminus \{G\}) = \mathcal{K}(q')$, we have $\mathcal{K}(q') \models \mathsf{KVars}(G) \rightarrow L$. By Lemma 4.3, $\zeta^+[L] = \mu[L]$. Let τ_G, τ_F be the two join trees obtained from τ by cutting the edge e with label L, where τ_G contains G, and τ_F contains F. This corresponds to the situation depicted in Figure 12. Let κ be the valuation over U such that for every $x \in U$,

$$\kappa(x) = \begin{cases} \mu(x) & \text{if } x \text{ occurs in } \tau_G \\ \zeta^+(x) & \text{if } x \text{ occurs in } \tau_F. \end{cases}$$

Notice that if x occurs in both τ_G and τ_F, then, by the *Connectedness Condition*, x occurs in L, hence $\mu(x) = \zeta^+(x)$. Obviously, $\kappa(q) \subseteq \mathbf{rep}$. It follows $\mathbf{rep} \models \zeta(q)$. □

The proof of Lemma 8.10 is given next.

Proof of Lemma 8.10. The proof of the first item is straightforward. We next prove the second item. From Lemma B.1 and by using induction on the length of the uniformization, we obtain:

$$\mathsf{Reify}(q, F, \mathbf{r}_n) \subseteq \mathsf{Reify}(q, F, \mathbf{r}), \text{ and} \tag{6}$$
$$\mathsf{Reify}(q, F, \mathbf{s}_n) \subseteq \mathsf{Reify}(q, F, \mathbf{s}). \tag{7}$$

We show $\mathsf{Reify}(q, F, \mathbf{r}_n) \subseteq \mathsf{Reify}(q, F, \mathbf{s}_n)$. Let θ be a valuation over $\mathsf{KVars}(F)$ such that $\mathbf{r}_n \models \theta(q)$. Let U be the set of variables that occur in q. We can assume a valuation θ^+ that extends θ to U such that $\theta^+(q) \subseteq \mathbf{r}_n$. Then, every atom in $\theta^+(q)$ is relevant for q in $\mathbf{r}_n$. By our uniformization construction, $\theta^+(q) \subseteq \mathbf{s}_n$. It follows $\mathbf{s}_n \models \theta(q)$. Consequently, $\mathsf{Reify}(q, F, \mathbf{r}_n) \subseteq \mathsf{Reify}(q, F, \mathbf{s}_n)$, hence $\mathsf{Reify}(q, F, \mathbf{r}_n) \subseteq \mathsf{Reify}(q, F, \mathbf{s})$ by (7). □

C. PROOF OF THEOREM 8.13

We make use of the following helping lemmas.

LEMMA C.1. *Let q be an acyclic Boolean conjunctive query with an acyclic attack graph. Let x be a variable that occurs in q. Let q' be the query obtained from q by replacing each occurrence of x with some constant c that does not occur in q. Then q' is an acyclic conjunctive query with an acyclic attack graph.*

PROOF. Let τ be a join tree for q. Let τ' be the graph obtained from τ by replacing, in all vertices, each occurrence of x with c (and by deleting x from each edge label). Obviously, τ' is a join tree for q'. It suffices to show that the attack graph of τ' is acyclic.

Let θ be the valuation defined by $\theta = \{x \mapsto c\}$. For every atom $F \in q$, let $F' = \theta(F)$. We first show that for every $F \in q$, we have $F^{+,q} \setminus \{x\} \subseteq (F')^{+,q'}$. The computation of the attribute closure $F^{+,q}$ by means of a standard algorithm [Abiteboul et al. 1995, page 165] corresponds to constructing a maximal sequence (see also the proof of Lemma 4.9)

$$\mathsf{KVars}(F) = \begin{array}{ll} S_0 & H_1 \\ S_1 & H_2 \\ \vdots & \vdots \\ S_{k-1} & H_k \\ S_k & \end{array}$$

where

(1) $S_0 \subsetneq S_1 \subsetneq \cdots \subsetneq S_{k-1} \subsetneq S_k$; and
(2) for every $i \in \{1, 2, \ldots, k\}$, we have $H_i \in q \setminus \{F\}$ such that $\mathsf{KVars}(H_i) \subseteq S_{i-1}$ and $S_i = S_{i-1} \cup \mathsf{Vars}(H_i)$.

Then, $S_k = F^{+,q}$. We can construct a corresponding sequence for q', as follows.

$$\mathsf{KVars}(F') = \begin{array}{ll} S_0 \setminus \{x\} & H'_1 \\ S_1 \setminus \{x\} & H'_2 \\ \vdots & \vdots \\ S_{k-1} \setminus \{x\} & H'_k \\ S_k \setminus \{x\} & \end{array}$$

Obviously, for every $i \in \{1, 2, \ldots, k\}$, we have

—$H'_i \in q' \setminus \{F'\}$,
—$\mathsf{KVars}(H'_i) \subseteq S_{i-1} \setminus \{x\}$, and
—$S_i \setminus \{x\} = (S_{i-1} \setminus \{x\}) \cup \mathsf{Vars}(H'_i)$.

This corresponding sequence shows $F^{+,q} \setminus \{x\} \subseteq (F')^{+,q'}$.

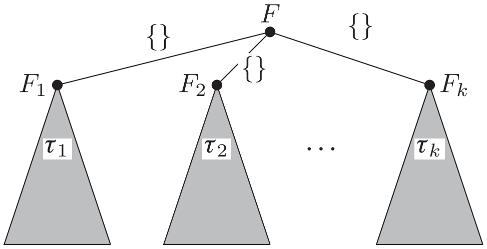


Fig. 13. Join tree τ_F in Lemma C.2.

Let F, G be distinct atoms of q such that $F \overset{\tau}{\not\rightsquigarrow} G$. Then, for some edge $H \overset{L}{\frown} J$ on the path between F and G in τ, we have $L \subseteq F^{+,q}$. Obviously, the path between F' and G' in τ' contains an edge $H' \overset{L'}{\frown} J'$ with $L' = L \setminus \{x\}$. From $L \setminus \{x\} \subseteq F^{+,q} \setminus \{x\}$ and $F^{+,q} \setminus \{x\} \subseteq (F')^{+,q'}$, it follows $L' \subseteq (F')^{+,q'}$. Consequently, $F' \overset{\tau'}{\not\rightsquigarrow} G'$. It follows that the attack graph of τ' must be acyclic. □

Lemma C.2. *Let q be an acyclic Boolean conjunctive query with an acyclic attack graph. Let F be a ground atom that belongs to q. Then $q \setminus \{F\}$ is an acyclic conjunctive query with an acyclic attack graph.*

Proof. Let τ be a join tree for q. Let τ_F be the directed rooted join tree obtained from τ by selecting F as the root. Let F_1, F_2, ..., F_k be the children of F in τ_F. For $i \in \{1, 2, \ldots, k\}$, let τ_i be the vertex-induced subtree of τ_F induced by F_i and all descendants of F_i. The situation is depicted in Figure 13. Clearly, if e is an edge adjacent to F, then the label of e is the empty set. Let τ' be the tree that contains τ_1, τ_2, ..., τ_k and additional edges $F_1 \overset{\{\}}{\frown} F_2$, $F_2 \overset{\{\}}{\frown} F_3$, ..., $F_{k-1} \overset{\{\}}{\frown} F_k$. It is straightforward that τ' is a join tree for $q \setminus \{F\}$ and that the attack graph of τ' is acyclic. □

The proof of Theorem 8.13 is given next.

Proof of Theorem 8.13. The proof runs by induction on the length of q. The result is obvious for $q = \{\}$. For the induction step, assume $q \neq \{\}$. Assume that the attack graph of q is acyclic. Let $R_1(\underline{\vec{x}_1}, \vec{y}_1)$, $R_2(\underline{\vec{x}_2}, \vec{y}_2)$, ..., $R_n(\underline{\vec{x}_n}, \vec{y}_n)$ be a topological sorting of the atoms of q with respect to the attack graph. $R_1(\underline{\vec{x}_1}, \vec{y}_1)$ is reifiable by Corollary 8.11.

Let $\vec{v} = \langle v_1, v_2, \ldots, v_n \rangle$ be a sequence of distinct variables containing each variable in $\mathsf{vars}(\vec{x}_1) \cup \mathsf{vars}(\vec{y}_1)$. Let $q'(\vec{v})$ be the nonBoolean conjunctive query whose set of atoms is $q \setminus \{R_1(\underline{\vec{x}_1}, \vec{y}_1)\}$. Let $\vec{c} = \langle c_1, c_2, \ldots, c_n \rangle$ be a sequence of n new constants. Let $q'_{\vec{v} \mapsto \vec{c}}$ be the query obtained from q' by replacing each v_i with c_i. It follows from Lemmas C.1 and C.2 that $q'_{\vec{v} \mapsto \vec{c}}$ is an acyclic conjunctive query with an acyclic attack graph. By the induction hypothesis, $q'_{\vec{v} \mapsto \vec{c}}$ has a certain first-order rewriting $\tilde{\varphi}$. Let $\varphi(\vec{v})$ be the formula obtained from $\tilde{\varphi}$ by replacing all occurrences of c_i with v_i $(1 \leq i \leq n)$. Since $\varphi(\vec{v})$ is a certain first-order rewriting for $q'(\vec{v})$, the desired result follows by Lemma 8.6. □

ACKNOWLEDGMENTS

Theorem 6.1 was brought to me by Alexandre Decan. Theorem 9.1 was pointed out to me by Phokion Kolaitis. We thank the anonymous referees for their insightful comments and suggestions.

REFERENCES

Abiteboul, S., Hull, R., and Vianu, V. 1995. *Foundations of Databases*. Addison-Wesley.

Andritsos, P., Fuxman, A., and Miller, R. J. 2006. Clean answers over dirty databases: A probabilistic approach. In *Proceedings of the International Conference on Data Engineering (ICDE'06)*. L. Liu, A. Reuter, K.-Y. Whang, and J. Zhang, Eds., IEEE Computer Society, 30.

Arenas, M., Bertossi, L. E., and Chomicki, J. 1999. Consistent query answers in inconsistent databases. In *Proceedings of the ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS'99)*. ACM Press, New York, 68–79.

Beeri, C. and Bernstein, P. A. 1979. Computational problems related to the design of normal form relational schemas. *ACM Trans. Datab. Syst. 4*, 1, 30–59.

Beeri, C., Fagin, R., Maier, D., and Yannakakis, M. 1983. On the desirability of acyclic database schemes. *J. ACM 30*, 3, 479–513.

Cayley, A. 1889. A theorem on trees. *Quart. J. Math. 23*, 376–378.

Chomicki, J. and Marcinkowski, J. 2005. Minimal-Change integrity maintenance using tuple deletions. *Inf. Comput. 197*, 1-2, 90–121.

Dalvi, N. N., Ré, C., and Suciu, D. 2009. Probabilistic databases: Diamonds in the dirt. *Comm. ACM 52*, 7, 86–94.

Dalvi, N. N., Ré, C., and Suciu, D. 2011. Queries and materialized views on probabilistic databases. *J. Comput. Syst. Sci. 77*, 3, 473–490.

Dalvi, N. N. and Suciu, D. 2007. Efficient query evaluation on probabilistic databases. *VLDB J. 16*, 4, 523–544.

Dasgupta, S., Papadimitriou, C. H., and Vazirani, U. V. 2008. *Algorithms*. McGraw-Hill.

Fagin, R., Kolaitis, P. G., Miller, R. J., and Popa, L. 2005. Data exchange: Semantics and query answering. *Theor. Comput. Sci. 336*, 1, 89–124.

Fuxman, A., Fazli, E., and Miller, R. J. 2005. Conquer: Efficient management of inconsistent databases. In *Proceedings of the ACM SIGMOD Conference on Management of Data*. F. Özcan, Ed., ACM Press, New York, 155–166.

Fuxman, A. and Miller, R. J. 2005. First-Order query rewriting for inconsistent databases. In *Proceedings of the International Conference on Database Technology (ICDT'05)*. T. Eiter and L. Libkin, Eds., Lecture Notes in Computer Science, vol. 3363, Springer, 337–351.

Fuxman, A. and Miller, R. J. 2007. First-Order query rewriting for inconsistent databases. *J. Comput. Syst. Sci. 73*, 4, 610–635.

Gottlob, G., Leone, N., and Scarcello, F. 2002. Hypertree decompositions and tractable queries. *J. Comput. Syst. Sci. 64,* 3, 579–627.

Grieco, L., Lembo, D., Rosati, R., and Ruzzi, M. 2005. Consistent query answering under key and exclusion dependencies: Algorithms and experiments. In *Proceedings of the Conference on Information and Knowledge Management (CIKM'05),* O. Herzog, H.-J. Schek, N. Fuhr, A. Chowdhury, and W. Teiken, Eds., ACM Press, New York, 792–799.

Huang, J., Antova, L., Koch, C., and Olteanu, D. 2009. MayBMS: A probabilistic database management system. In *Proceedings of the ACM SIGMOD Conference on Management of Data.* U. Çentintemel, S. B. Zdonik, D. Kossman, and N. Tatbul, Eds., ACM Press, New York, 1071–1074.

Kolaitis, P. G. and Pema, E. 2012. A dichotomy in the complexity of consistent query answering for queries with two atoms. *Inf. Process. Lett. 112*, 3, 77–85.

Lembo, D., Rosati, R., and Ruzzi, M. 2006. On the first-order reducibility of unions of conjunctive queries over inconsistent databases. In *Proceedings of the EDBT Workshops*, T. Grust, H. Höpfner, A. Illarramendi, S. Jablonski, M. Mesiti, S. Müller, P.-L. Patranjan, K.-U. Sattler, M. Spiliopoulou, and J. Wijsen, Eds., Lecture Notes in Computer Science, vol. 4254, Springer, 358–374.

Libkin, L. 2004. *Elements of Finite Model Theory*. Springer.

Maier, D. 1983. *The Theory of Relational Databases*. Computer Science Press.

Maslowski, D. and Wijsen, J. 2011. On counting database repairs. In *Proceedings of the National Low Impact Development Conference (LID'11)*. G. H. L. Fletcher and S. Staworko, Eds., ACM Press, New York, 15–22.

Ullman, J. D. 1988. *Principles of Database and Knowledge-Base Systems* Vol. I. Computer Science Press.

Wijsen, J. 2009a. Consistent query answering under primary keys: A characterization of tractable queries. In *Proceedings of the International Conference on Database Technology (ICDT'09)*. R. Fagin, Ed., ACM International Conference Proceedings, vol. 361, ACM Press, New York, 42–52.

Wijsen, J. 2009b. On the consistent rewriting of conjunctive queries under primary key constraints. *Inf. Syst. 34*, 7, 578–601.

WIJSEN, J. 2010a. On the first-order expressibility of computing certain answers to conjunctive queries over uncertain databases. In *Proceedings of the ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS'10).* J. Paredaens and D. V. Gucht, Eds., ACM Press, New York, 179–190.

WIJSEN, J. 2010b. A remark on the complexity of consistent conjunctive query answering under primary key violations. *Inf. Process. Lett. 110*, 21, 950–955.

Received May 2011; revised October 2011; accepted December 2011