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 \( \text{CERTAINTY}(q) \) takes as input a database \( \text{db} \) and asks whether \( q \) evaluates to true on every repair of \( \text{db} \). We are interested in determining queries \( q \) for which \( \text{CERTAINTY}(q) \) is first-order expressible (and hence in the low complexity class \( \text{AC}^0 \)).

For queries \( q \) in the class of conjunctive queries without self-join, we provide a necessary syntactic condition for first-order expressibility of \( \text{CERTAINTY}(q) \). 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 \( \text{CERTAINTY}(q) \) when \( q \) is acyclic and without self-join. We also show that if \( \text{CERTAINTY}(q) \) 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

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 Fig. 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 Fig. 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.
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:

\[ q_1 = \exists x \exists y (R(\text{VLDB}', x, y) \land T(y, \text{Italy})), \]

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 \( \varphi_1 \) evaluates to true on the original database:

\[
\varphi_1 = \exists x \exists y \left( R(\text{VLDB}', x, y) \land \forall y \left( R(\text{VLDB}', x, y) \rightarrow \left( T(y, \text{Italy}) \land \forall z (T(y, z) \rightarrow z = \text{Italy}) \right) \right) \right).
\]

Formally, a certain first-order rewriting for a Boolean query \( q \) is a first-order sentence \( \varphi \) such that for every database \( db \), \( q \) evaluates to true on every repair of \( db \) if and only if \( \varphi \) 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 \( \varphi \) once on the original database. Since \( \varphi \) 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 conjunctive query:

\[
\text{CERTAINTY}(q) = \{ db \mid q \text{ evaluates to true on every repair of } db \}.
\]

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

We study the decidability of first-order expressibility of \( \text{CERTAINTY}(q) \) 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 \( \text{CERTAINTY}(q) \) 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 \( \text{CERTAINTY}(q) \).
(2) Find a syntactic characterization of the frontier between tractable and intractable cases of CERTAINTY(q).

Some partial results are known. The problem CERTAINTY(q) is obviously in coNP for first-order queries q, and it is known that CERTAINTY(q2) is coNP-complete for \( q_2 = \exists x \exists y \exists z (R(x, z) \land S(y, z)) \) [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 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 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 CERTAINTY(q) when q is acyclic and without self-join. Section 9 settles the complexity of determining first-order expressibility of 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 \( \text{vars}(\vec{x}) \) 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 \( \theta \) from \( U \) to the set of constants. Such valuation \( \theta \) is often extended to be the identity on constants and on variables not in \( U \).

Key-equivalent 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 \( R(\vec{x}, \vec{y}) \) 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 \( R_1(\vec{a}_1, \vec{b}_1), R_2(\vec{a}_2, \vec{b}_2) \) are key-equivalent 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 \( \mathcal{DB} \) of ground atoms using only the relation names of the schema. Importantly, a database can contain distinct, key-equivalent atoms. Intuitively, if a database contains distinct, key-equivalent 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 \( q = \{ R_1(x_1, y_1), \ldots, R_n(x_n, y_n) \} \) of atoms. 1 This set represents the first-order sentence \( \exists u_1 \ldots \exists u_k (R_1(x_1, y_1) \land \cdots \land R_n(x_n, y_n)) \) where \( u_1, \ldots, u_k \) are all the variables that occur in \( x_1, y_1, \ldots, x_n, y_n \). This query \( q \) is satisfied by a database db, denoted \( db \models q \), if there exists a valuation \( \theta \) over \( vars(x_1, y_1, \ldots, x_n, y_n) \) such that for each \( i \in \{1, \ldots, n\} \), \( R_i(\theta(x_i), \theta(y_i)) \in db \). 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 non-Boolean 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 db a database. We write \( db \models^\varphi q \) if for every repair \( rep \) of db, we have \( rep \models q \). Given a Boolean conjunctive query \( q \), CERTAINTY\((q)\) is (the complexity of) the following set:

\[
\text{CERTAINTY}(q) = \{ db \mid db \text{ is a database and } db \models^\varphi q \}.
\]

CERTAINTY\((q)\) is said to be first-order expressible if there exists a first-order sentence \( \varphi \) such that for every database \( db \subseteq \text{CERTAINTY}(q) \) if and only if \( db \models \varphi \). The formula \( \varphi \), 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 \( F = R(\vec{x}, \vec{y}) \), we denote by KVars\((F)\) the set of variables that occur in \( \vec{x} \) and by \( \text{Vars}(F) \) the set of variables that occur in \( F \), that is, \( \text{KVars}(F) = \text{vars}(\vec{x}) \) and \( \text{Vars}(F) = \text{vars}(\vec{x}) \cup \text{vars}(\vec{y}) \).

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 \( \text{Vars}(F) \cap \text{Vars}(G) \). 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 \( \tau \) will be used for join trees, and \( \rho \) for intersection trees.

We write \( F \bowtie L G \) to denote an edge between \( F \) and \( G \) with label \( L \).

A join tree is shown in Fig. 2 (left). The query \( \{ R_0(\vec{z}, u), R_1(\vec{z}, y), R_2(\vec{z}, x, u) \} \) is cyclic and hence has no join tree. An intersection tree for that query is shown in Fig. 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 rewritings of conjunc-

---

1Up to Section 8, all queries are understood to be Boolean and quantifiers are omitted.
tive 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 $C_{forest}$, such that every query in that class has a certain first-order rewriting. At the same time, however, they recognized that the query $q_3 = \exists x \exists y (R(x, y) \land S(x, y))$ is not in $C_{forest}$ 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 $C_{forest}$.

Relatively little is known about CERTAINTY($q$) for conjunctive queries $q$ that are cyclic and/or contain self-joins. For the query $q_4 = \exists x \exists y (R(x, y) \land R(y, a))$, with self-join, it is known that CERTAINTY($q_4$) is in 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 $m \geq 2$:

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

which are cyclic if $m \geq 3$. Unfortunately, these claims are based on incorrect arguments, as remarked by Wijsen [2010b], who showed that for $q_5 = \exists x \exists y (R_1(x, y) \land R_2(y, x))$ (that is, for $m = 2$), it is the case that CERTAINTY($q_5$) is in P, albeit not first-order expressible.

In [Wijsen 2009b], Wijsen defined a semantic class $C_{rooted}$ of conjunctive queries, and showed first-order expressibility of CERTAINTY($q$) for all $q \in C_{rooted}$. Queries in $C_{rooted}$ can be cyclic and contain self-joins. However, no membership test for $C_{rooted}$ 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 $C_{rooted}$.

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

Maslowski and Wijsen [2011] have studied the complexity of the counting variant of CERTAINTY($q$), denoted $\sharp$CERTAINTY($q$). Given a database $db$, the problem $\sharp$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: $\sharp$CERTAINTY($q$) is in P or $\sharp$P-complete. The problem $\sharp$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; Dalvi et al. 2011]. A block in a database $db$ is a maximal subset of key-equal atoms. If $\{R(\bar{a}, b_1), \ldots, R(\bar{a}, b_n)\}$ 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. Greco 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.

Every atom \( F \) in a query \( q \) gives rise to a functional dependency among the variables that occur in \( F \). For example, \( R(x, y, z) \) 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:

\[
\mathcal{K}(q) = \{ \text{KVars}(F) \rightarrow \text{Vars}(F) \mid F \in q \}.
\]

**Example 4.2.** Let \( q_6 \) denote the query whose join tree \( \tau_6 \) is shown in Fig. 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 \( R_2(a, x) \) whose primary key value contains no variables.

**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 \( \theta, \mu \) 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 \( R(x, y) \) of \( q \) such that \( X = \text{vars}(x) \) and \( Y = \text{vars}(\bar{x}) \cup \text{vars}(\bar{y}) \). Assume

\[
\theta(x) = \alpha, \quad \theta(y) = \beta, \quad \mu(x) = \gamma, \quad \mu(y) = \delta.
\]

Then, \( \{x\} \rightarrow \{x, y\} \) is satisfied by \( \theta \) and \( \mu \) if \( \alpha = \beta \). On the other hand, \( \{x\} \rightarrow \{x, y\} \) is not satisfied by \( \theta \) and \( \mu \) if \( \alpha \neq \beta \).
\( \theta[X] = \mu[X]. \) We obviously have \( \theta(\bar{x}) = \mu(\bar{x}). \) From \( \theta(q), \mu(q) \subseteq \text{db}, \) it follows that \( R(\theta(\bar{x}), \theta(\bar{y})), R(\mu(\bar{x}), \mu(\bar{y})) \in \text{db}. \) Since \( \theta(\bar{x}) = \mu(\bar{x}) \) and since \( \text{db} \) contains no two distinct key-equal atoms, we have \( \theta(\bar{y}) = \mu(\bar{y}). \) It follows \( \theta[Y] = \mu[Y]. \)

Concerning the following definition, recall from relational database theory [Ullman 1988, page 387] that if \( \Sigma \) is a set of functional dependencies over a set \( \mathcal{U} \) of attributes and \( X \subseteq \mathcal{U}, \) then the attribute closure of \( X \) (with respect to \( \Sigma \)) is the set \( \{ A \in \mathcal{U} | \Sigma \models X \rightarrow A \}. \) 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 \( \mathcal{U} \) be the set of variables that occur in \( q. \) For every \( F \in q, \) we define:

\[
F^{+q} = \{ x \in \mathcal{U} | K(q \setminus \{ F \}) \models K\text{Vars}(F) \rightarrow x \}.
\]

In words, \( F^{+q} \) is the attribute closure of the set \( K\text{Vars}(F) \) 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 an attack graph will turn out to be a powerful tool for characterizing first-order expressibility of \( \text{CERTAINTY}(q). \) In particular, we will show that the following properties hold for all Boolean conjunctive queries \( q \) with self-join:

- If \( \rho \) is an intersection tree for \( q \) such that the attack graph of \( \rho \) has a cycle with exactly two vertices, then \( \text{CERTAINTY}(q) \) 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 \( \text{CERTAINTY}(q) \) is not first-order expressible. This is Theorem 7.4.
- If \( q \) is acyclic and the attack graph of \( q \) is acyclic, then \( \text{CERTAINTY}(q) \) 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 \( \rho \) be an intersection tree for Boolean conjunctive query \( q. \) The **attack graph** of \( \rho \) 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 \( \rho, \) we have \( L \notin F^{+q}. \)

We write \( F \overset{L}{\leftarrow} G \) if the attack graph of \( \rho \) contains a directed edge from \( F \) to \( G. \) If \( F \overset{L}{\leftarrow} G, \) 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 \( F_0 \overset{L_1}{\leftarrow} F_1 \overset{L_2}{\leftarrow} F_2 \cdots \overset{L_{n-1}}{\leftarrow} F_{n-1} \overset{L_n}{\leftarrow} F_0. \)

**Example 4.6.** Consider again join tree \( \tau_0 \) for query \( \rho_0 \) in Fig. 2 (left). To shorten notation, let \( F = R_0(\bar{x}, y), \) \( G = R_1(y, x), \) and \( H = R_2(\bar{x}, x), \) as indicated in Fig. 2 (left). The attack graph of \( \tau_0 \) is shown in Fig. 2 (right) and is computed as follows. We have:

\[
K(q_0 \setminus \{ F \}) = \{ \{ y \} \rightarrow \{ x, y \}, \{ \} \rightarrow \{ x \} \}
\]

\[
K(q_0 \setminus \{ G \}) = \{ \{ x \} \rightarrow \{ x, y \}, \{ \} \rightarrow \{ x \} \}
\]

\[
K(q_0 \setminus \{ H \}) = \{ \{ x \} \rightarrow \{ x, y \}, \{ y \} \rightarrow \{ x, y \} \}
\]
We have $\text{KVars}(F) = \{x\}$, which is already closed with respect to $\mathcal{K}(q_0 \setminus \{F\})$. Thus, $F^{+ \cdot q_0} = \{x\}$. The path from $F$ to $G$ in the join tree is $F \not\rightarrow G$. Since the label $\{x, y\}$ is not contained in $F^{+ \cdot q_0}$, the attack graph contains a directed edge from $F$ to $G$, i.e., $F \xrightarrow{\tau_6} G$. The path from $F$ to $H$ in the join tree is $F \not\rightarrow H$. Since the label $\{x\}$ is contained in $F^{+ \cdot q_6}$, the attack graph contains a directed edge from $F$ to $H$.

We have $\text{KVars}(G) = \{y\}$ and the closure of $\{y\}$ with respect to $\mathcal{K}(q_0 \setminus \{G\})$ is $\{x, y\}$.

Thus, $G^{+ \cdot q_0} = \{x, y\}$. The path from $G$ to $F$ in the join tree is $G \not\rightarrow F$. Since the label $\{x, y\}$ is contained in $G^{+ \cdot q_6}$, 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 $\text{KVars}(H) = \{\}$, which is already closed with respect to $\mathcal{K}(q_0 \setminus \{H\})$.

Thus, $H^{+ \cdot q_0} = \{\}$. The path from $H$ to $G$ in the join tree is $H \not\rightarrow F \not\rightarrow G$. Since no label on that path is contained in $H^{+ \cdot q_6}$, the attack graph contains a directed edge from $H$ to $G$, i.e., $H \xrightarrow{\tau_6} G$. It is then obvious that the attack graph must also contain a directed edge from $H$ to $F$, i.e., $H \xrightarrow{\tau_6} F$.

**Example 4.7.** Consider join tree $\tau_7$ for query $q_7$ in Fig. 3 (left). Notice that $R_2$ is all-key. We have:

$$\mathcal{K}(q_7 \setminus \{F\}) \equiv \{y \rightarrow x, x \rightarrow z\}$$

$$\mathcal{K}(q_7 \setminus \{G\}) \equiv \{x \rightarrow y, x \rightarrow z\}$$

$$\mathcal{K}(q_7 \setminus \{H\}) = \mathcal{K}(q_7 \setminus \{J\}) = \mathcal{K}(q_7 \setminus \{K\}) \equiv \{x \rightarrow y, y \rightarrow x, x \rightarrow z\}$$

Consequently,

$$F^{+ \cdot q_7} = \{x, z\}$$

$$G^{+ \cdot q_7} = \{y\}$$

$$H^{+ \cdot q_7} = \{x, y, z\}$$

$$J^{+ \cdot q_7} = \{x, y, z\}$$

$$K^{+ \cdot q_7} = \{x, y, z\}$$

Since no edge label is contained in $G^{+ \cdot q_7}$, the atom $G$ attacks every other atom. The complete attack graph is shown in Fig. 3 (right). It contains a cycle $F \xrightarrow{\tau_6} G \xrightarrow{\tau_6} F$ of size 2. Notice that $F \xrightarrow{\tau_7} G$ and $G \xrightarrow{\tau_6} J$, but $F \xrightarrow{\tau_6} J$. So attack graphs need not be transitive.

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

**Lemma 4.9.** Let $q$ be a Boolean conjunctive query. Let $\rho$ be an intersection tree for $q$. Let $F, G$ be distinct atoms of $q$.

1. If $F \searrow G$, then $\text{KVars}(G) \not\subseteq F^{+ \cdot q}$.
2. If $F \searrow G$ and $H \subseteq [F, G]_{\rho} \setminus \{F\}$, then $F \not\rightarrow H$.
3. If $F \not\rightarrow G$, then $\mathcal{K}(q \setminus [F, G]_{\rho}) \models \text{KVars}(F) \rightarrow F^{+ \cdot q}$.

**Proof.**

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

(2) Straightforward.

(3) Assume $F \not\models G$. The computation of the attribute closure $F^{+}$ by means of a standard algorithm [Abiteboul et al. 1995, page 165] corresponds to constructing a maximal sequence:

\[
\text{KVars}(F) = S_0 \quad H_1 \quad S_1 \quad H_2 \quad \vdots \quad \vdots \quad S_{k-1} \quad H_k \quad S_k
\]

where

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

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

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

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

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

$\square$

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 $\{R(z, y), S(y, x)\}$ appears in [Wijsen 2010b].

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

**Proof.** Let $U$ be the set of variables that occur in $q$. Let $\rho$ 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 $F \not\models G$ and $G \not\models F$.

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

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

- $\text{db}_{no} \not\in \text{CERTAINTY}(q)$;
- $\text{db}_{yes} \in \text{CERTAINTY}(q)$; and
This leads to a contradiction that concludes the proof. The following sequences of variables are defined in the construction:

- Let $\vec{x}$ be the variables (in some fixed order) of $F^+ \setminus G^+ \setminus q \setminus G^+ \setminus q$. Obviously, $\text{KVars}(F) \not\subseteq F^+ \setminus q$. By item (1) in Lemma 4.9, $\text{KVars}(F) \not\subseteq G^+ \setminus q$. It follows that $\vec{x}$ contains at least one variable.

- Let $\vec{z}$ be the variables of $G^+ \setminus q \setminus G^+ \setminus q$.

- Let $\vec{y}$ be the variables of $F^+ \setminus q \cap G^+ \setminus q$.

- Let $\vec{u}$ contain all other variables, i.e., $\vec{u}$ contains all variables in $(U \setminus F^+ \setminus q) \setminus G^+ \setminus q$.

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 Fig. 4. The intended meaning is that distinct letters and/or subscripts denote distinct constants. The boxes in Fig. 4 highlight where constants are reused in different valuations. In particular, for all $i, j \in \{1, 2, \ldots, 2m + 1\}$, for all $v \in U$, $\theta_i(v) = \mu_j(v) \iff v \in \text{vars}(\vec{y})$. Moreover, for all $i, j \in \{1, 2, \ldots, 2m + 1\}$ such that $i < j$, for all $v \in U \setminus \text{vars}(\vec{y})$,

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

\begin{table}[h]
\centering
\begin{tabular}{cccc}
\hline
 & $a_1$ & $b$ & $c_1$ & $d_1$ & $\theta_1$ \\
\hline
$a_2$ & $b$ & $c_2$ & $d_2$ & $\theta_2$ \\
$d_3$ & $\theta_3$ \\
$a_3$ & $b$ & $c_3$ & $d_3$ & $\theta_4$ \\
$d_4$ & $\theta_4$ \\
$a_3$ & $b$ & $c_3$ & $d_4$ & $\theta_5$ \\
$d_5$ & $\theta_5$ \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
$a_m$ & $b$ & $c_m$ & $d_{2m - 1}$ & $\theta_{2m - 1}$ \\
$c_m$ & $\mu_1$ \\
$a_{m + 1}$ & $b$ & $c_{m + 1}$ & $d_{2m}$ & $\theta_{2m}$ \\
$c_{m + 1}$ & $\mu_2$ \\
$a_{m + 1}$ & $b$ & $c_{m + 1}$ & $d_{2m + 1}$ & $\theta_{2m + 1}$ \\
$c_{m + 1}$ & $\mu_3$ \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
$c_m$ & $\bar{g}_m$ & $\bar{h}_m$ & $\bar{h}_{2m - 1}$ & $\mu_{2m - 1}$ \\
$c_{m + 1}$ & $\bar{g}_{m + 1}$ & $\bar{h}_{2m}$ & $\mu_{2m}$ \\
$c_{m + 1}$ & $\bar{g}_{m + 1}$ & $\bar{h}_{2m + 1}$ & $\mu_{2m + 1}$ \\
\hline
\end{tabular}
\caption{Valuations used in the construction of $db_{\text{yes}}$ and $db_{\text{no}}$.}
\end{table}
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.

--- $v \in \text{vars}(\bar{x})$ and $i$ is even; or
--- $v \in \text{vars}(\bar{z})$ and $i$ is odd.

Let

$$db_\theta = \bigcup \{ \theta_i(q) \mid 1 \leq i \leq 2m + 1 \}$$
$$db_\mu = \bigcup \{ \mu_i(q) \mid 1 \leq i \leq 2m + 1 \}$$
$$db = db_\theta \cup db_\mu$$

Notice that the only constants common to $db_\theta$ and $db_\mu$ occur in $\bar{b} = \theta_1(\bar{y})$. 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 Fig. 5. For Property 5.4, say that two distinct atoms $A, B \in db$ are joinable (with respect to $q$) if $A, B \in \omega(q) \subseteq db$ for some valuation $\omega$ over $U$. Then, Property 5.4 expresses that two distinct atoms in Fig. 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 \( K\text{Vars}(F) \subseteq F^{+q} \) 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 \( F \not\subseteq G \), it follows \( L \not\subseteq F^{+q} \). We can assume a variable \( u \in L \setminus F^{+q} \). Since \( u \in L \), we have \( u \in \text{Vars}(F) \). Since \( u \not\in F^{+q} \), 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 \( A, B \in \text{db} \), 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 \( i \in \{1, 2, \ldots, 2m\} \), \( \{A, B\} = \{\mu_i(G), \mu_{i+1}(G)\} \);
3. for some odd number \( i \in \{1, 2, \ldots, 2m\} \), \( \{A, B\} = \{\theta_i(G), \theta_{i+1}(G)\} \);
4. for some odd number \( i \in \{1, 2, \ldots, 2m\} \), \( \{A, B\} = \{\mu_i(G), \mu_{i+1}(G)\} \).

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

We show that \( H = F \) or \( H = G \). Assume, towards a contradiction, \( H \neq F \) and \( H \neq G \). Since \( K(q \setminus \{F\}) \) contains the functional dependency \( K\text{Vars}(H) \rightarrow \text{Vars}(H) \), we have \( \text{Vars}(H) \subseteq F^{+q} \). By our construction, the only possible way of having two distinct key-equal atoms is then \( K\text{Vars}(H) \subseteq \text{vars}(\tilde{y}) \) and \( \text{Vars}(H) \cap \text{vars}(\tilde{x}) \neq \{\} \). From \( K\text{Vars}(H) \subseteq \text{vars}(\tilde{y}) \), it follows \( K\text{Vars}(H) \subseteq G^{+q} \). Since \( K(q \setminus \{G\}) \) contains the functional dependency \( K\text{Vars}(H) \rightarrow \text{Vars}(H) \), we have \( \text{Vars}(H) \subseteq G^{+q} \). Since \( \text{Vars}(H) \subseteq F^{+q} \), it follows \( \text{Vars}(H) \subseteq \text{vars}(\tilde{y}) \), hence \( \text{Vars}(H) \cap \text{vars}(\tilde{x}) = \{\} \), 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 \( G \not\subseteq F \), we have \( K\text{Vars}(F) \not\subseteq G^{+q} \) by item (1) in Lemma 4.9. Hence, we can assume a variable \( x \in K\text{Vars}(F) \cap \text{vars}(\tilde{x}) \). 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 \( \{A, B\} = \{\mu_i(G), \mu_{i+1}(G)\} \).

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 Fig. 5.

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

**Proof.** Let \( \omega \) be a valuation over \( U \) such that \( \omega(q) \subseteq \text{db} \). 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 \( H \not\rightarrow J \) on the path between \( F \) and \( G \) in \( \rho \), we have \( \omega(H) = \theta_i(H) \) and \( \omega(J) = \zeta(J) \) for some \( \zeta \in I \). Since \( F \not\subseteq G \) and \( G \not\subseteq F \), we have \( L \not\subseteq F^{+q} \) and \( L \not\subseteq G^{+q} \). Two cases can occur:
The difference between $\db_{\yes}$ and $\db_{\no}$ is in the atoms having the same relation name as $F$ or $G$ in $\db_{\yes}$ (left) and in $\db_{\no}$ (right). Conflicting, key-equal atoms are linked by a double-arrowed edge.

(1) $L \nsubseteq F^{+q} \cup G^{+q}$. We can assume $u \in L$ such that $u \in \text{vars}(\bar{u})$. Clearly, $u \in \text{Vars}(H)$ and $u \in \text{Vars}(J)$. 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, $x \in \text{vars}(\bar{x})$ and $z \in \text{vars}(\bar{z})$. 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 $\theta_i = \zeta$.

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 $\omega$ over $U$, $\omega(q) \subseteq \db$ implies that for some $\zeta \in I$, $\omega(F) = \zeta(F)$ and $\omega(G) = \zeta(G)$. □

The databases $\db_{\yes}$ and $\db_{\no}$ are defined as follows:

$$\db_{\yes} = \db \setminus \{\mu_1(F), \mu_{2m+1}(G)\}$$
$$\db_{\no} = \db \setminus \{\mu_1(F), \theta_{2m+1}(G)\}$$

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

(1) $\db_{\no} \notin \text{CERTAINTY}(q)$. Let rep be a repair of $\db_{\no}$ 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 $\db_{\no}$ in Fig. 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, $\neg \frac{q}{\text{rep}}$. 

Fig. 6. Case $m = 2$. The figure shows the atoms having the same relation name as $F$ or $G$ in $\db_{\yes}$ (left) and in $\db_{\no}$ (right). Conflicting, key-equal atoms are linked by a double-arrowed edge.
the choice of \( m \). Assume that there are no constant positions (in particular, \( G \) hereafter, cannot be used for distinguishing \( t \) and \( s \)). It can be easily seen that for every repair \( \text{rep} \) of \( \text{db}_{yes} \), there exists \( i \in \{1, 2, \ldots, 2n+1\} \) such that \( b_i(q) \subseteq \text{rep} \). Intuitively, with respect to \( \text{db}_{yes} \) in Fig. 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 \( \text{db}_{yes} \) and \( \text{db}_{no} \) are indistinguishable by \( \psi \). 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 \( \text{db}_{yes} \) agree on position \( i \) (likewise for \( \text{db}_{no} \)). Such positions, called constant positions hereafter, cannot be used for distinguishing \( \text{db}_{yes} \) and \( \text{db}_{no} \). In the following we assume that there are no constant positions (in particular, \( \vec{y} \) is empty).

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

It is easy to verify that \( d_I(\bar{a}_i \bar{c}_i \bar{d}_i, \bar{a}_{m+1} \bar{c}_{m+1} \bar{d}_{2m+1}) = d_I(\bar{c}_i, \bar{a}_{m+1}) = 2m - 1 \). Assume constant \( k \) such that \( d_I(k, \bar{a}_i \bar{c}_i \bar{d}_i) \leq d_I \), where \( d_I \) is the Hanf-locality rank of \( \psi \).

By the triangle inequality, \( d_I(k, \bar{a}_{m+1} \bar{c}_{m+1} \bar{d}_{2m+1}) \geq 2m - 1 - d_I \). From \( m > d_I \) (see the choice of \( m \) at the beginning of the current proof), it follows \( 2m - 1 > 2d_I \), hence \( d_I(k, \bar{a}_{m+1} \bar{c}_{m+1} \bar{d}_{2m+1}) > d_I \).

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 \( \tau \) of \( d \)-neighborhood has the same number of constants realizing \( \tau \) in \( \text{db}_{yes} \) and \( \text{db}_{no} \) (intuitively, in Fig. 6, Greek letters denote positions in \( \text{db}_{yes} \) and \( \text{db}_{no} \) that are indistinguishable by \( \psi \); the important thing to notice is that \( \beta \) and \( \delta \) are swapped):

**Constants close to \( \alpha \).** Every constant \( k \) such that \( d_I(k, \bar{a}_i \bar{c}_i \bar{d}_i) \leq d_I \) realizes the same isomorphism type of \( d \)-neighborhood in \( \text{db}_{yes} \) and \( \text{db}_{no} \).

**Constants close to \( \beta \).** Let \( k \) and \( l \) be homologous constants such that \( d_I(k, \bar{a}_{m+1} \bar{c}_{m+1} \bar{d}_{2m+1}) = d_I(l, \bar{c}_{m+1} \bar{g}_{m+1} \bar{h}_{2m+1}) \leq d_I \). Then, the \( d \)-neighborhood of \( k \) in \( \text{db}_{yes} \) is isomorphic to the \( d \)-neighborhood of \( l \) in \( \text{db}_{no} \).

**Constants close to \( \gamma \).** Every constant \( k \) such that \( d_I(k, \bar{c}_i \bar{g}_i \bar{h}_i) \leq d_I \) realizes the same isomorphism type of \( d \)-neighborhood in \( \text{db}_{yes} \) and \( \text{db}_{no} \).

**Constants close to \( \delta \).** Let \( k \) and \( l \) be homologous constants such that \( d_I(k, \bar{c}_{m+1} \bar{g}_{m+1} \bar{h}_{2m+1}) = d_I(l, \bar{a}_{m+1} \bar{c}_{m+1} \bar{d}_{2m+1}) \leq d_I \). Then, the \( d \)-neighborhood of \( k \) in \( \text{db}_{yes} \) is isomorphic to the \( d \)-neighborhood of \( l \) in \( \text{db}_{no} \).
Finally, two homologous constants not referred to in the preceding items (i.e., constants far removed from the positions $\alpha$, $\beta$, $\gamma$, $\delta$) realize the same isomorphism type of $d$-neighborhood in $\text{db}_{yes}$ and $\text{db}_{no}$. It follows $\text{db}_{yes} \equiv_d \text{db}_{no}$. Since $d$ is the Hanf-locality rank of $\psi$, it follows $\text{db}_{yes} \models \psi \iff \text{db}_{no} \models \psi$.

Technically, the absence of constant positions is important in the above application of Hanf-locality. Indeed, assume that all $R$-atoms of $\text{db}_{yes}$ agree on position $i$, that is, $R(s_1, \ldots, s_r), R(t_1, \ldots, t_r) \in \text{db}_{yes}$ implies $s_i = t_i$ ($1 \leq i \leq r$). Then, for every constant $a$ that occurs in some $R$-atom of $\text{db}_{yes}$, the $2$-neighborhood of $a$ includes all $R$-atoms of $\text{db}_{yes}$ (likewise for $\text{db}_{no}$). To show that the theorem remains valid in the presence of constant positions, we reason as follows. Let $k$ be the quantifier rank of $\psi$. Our results so far imply that in the absence of constant positions, the value of $m$ can be chosen sufficiently large such that the duplicator has a winning strategy in a $k$-round Ehrenfeucht-Fraïssé game on $\text{db}_{yes}$ and $\text{db}_{no}$. 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 $q_8 = \{ R_0(y, z, u), R_1(z, y), R_2(z, x, u) \}$. 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 Fig. 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.
By item (2) in Lemma 4.9, 

\[ F \rightarrow q \]

Since \( F \rightarrow q \), called witness databases.

The precedence between edges in the attack graph and the existence of particular databases, \( F \rightarrow q \) containing distinct atoms \( F = R(\bar{x}, \bar{y}) \) and \( G = S(\bar{u}, \bar{w}) \). Let \( U \) be the set of variables that occur in \( q \). A witness database for \( (F, G) \) is a database \( db \) with two distinct, key-equal

**THEOREM 6.1.** Let \( q \) be an acyclic Boolean conjunctive query. Let \( \tau_1 \) and \( \tau_2 \) be two join trees for \( q \). The attack graphs of \( \tau_1 \) and \( \tau_2 \) are identical.

**PROOF.** Let \( F, G \) be distinct atoms of \( q \) such that \( F \not\rightarrow q \). We show \( F \not\rightarrow q \). 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 \( \tau_1 \) contains an edge between \( F \) and \( G \). Since \( F \not\rightarrow q \), we can assume a variable \( x \in \text{Vars}(F) \cap \text{Vars}(G) \) such that \( x \not\in F^{+\cdot} \). By the Connectedness Condition, the variable \( x \) is an element of every label on the unique path between \( F \) and \( G \) in \( \tau_1 \). It follows \( F \not\rightarrow q \).

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

- Since \( F \not\rightarrow q \), we can assume a variable \( x \in \text{Vars}(H) \cap \text{Vars}(G) \) such that \( x \not\in F^{+\cdot} \).
  By the Connectedness Condition, the variable \( x \) is an element of every label on the unique path between \( H \) and \( G \) in \( \tau_2 \).

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

Consequently, every label on the unique path between \( F \) and \( G \) in \( \tau_2 \) contains a variable that does not belong to \( F^{+\cdot} \). It follows \( F \not\rightarrow q \).

By symmetry, for all distinct atoms \( F, G \) of \( q \), \( F \not\rightarrow q \) implies \( F \not\rightarrow q \). Thus, the attack graphs of \( \tau_1 \) and \( \tau_2 \) contain the same directed edges. This concludes the proof. \( \Box \)

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 \( \tau \) for \( q \).

We will write \( F \not\rightarrow q \) if the attack graph of \( q \) contains a directed edge from \( F \) to \( G \). If \( q \) is clear from the context, we write \( F \leadsto G \) instead of \( F \not\rightarrow q \).

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 \( F = R(\bar{x}, \bar{y}) \) and \( G = S(\bar{u}, \bar{w}) \). Let \( U \) be the set of variables that occur in \( q \). A **witness database** for \( (F, G) \) is a database \( db \) with two distinct, key-equal

\[ \begin{align*}
R_1(\bar{x}, \bar{y}) & \quad R_0(\bar{y}, \bar{z}, \bar{u}) \\
\{x\} & \quad \{z, u\} \\
R_2(\bar{x}, x, u) & \quad R_2(\bar{x}, x, u)
\end{align*} \]

Fig. 9. Another intersection tree \( \rho' \) (left) and attack graph (right) for the cyclic query \( q_0 \). The attack graph is acyclic. Notice that \( \rho' \) is not a join tree.
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. $db$ has exactly two repairs, namely $db \setminus \{A_1\}$ and $db \setminus \{A_2\}$, both satisfying $q$;
2. there exists a valuation $\theta_1$ over $U$ such that $A_1, B_1 \in \theta_1(q) \subseteq db$;
3. there exists a valuation $\theta_2$ over $U$ such that $A_2, B_2 \in \theta_2(q) \subseteq db$; and
4. there exists no valuation $\mu$ over $U$ such that $A_1, B_2 \in \mu(q) \subseteq db$ or $A_2, B_1 \in \mu(q) \subseteq db$.

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 Fig. 2, a witness database for $\{R_0(\xi, \eta), R_1(\eta, x)\}$ is $\{R_0(\xi, 1), R_0(\xi, 2), R_1(1, \alpha), R_1(2, \alpha), R_2(\eta, \alpha)\}$.

**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 $q$.

1. $\Rightarrow$ Let $\theta_1, \theta_2$ be valuations over $U$ such that for every $x \in U, \theta_1(x) = \theta_2(x)$ if and only if $x \in F^+$. Let $db = \theta_1(q) \cup \theta_2(q)$. 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 $db$ is a witness database for $(F, G)$, by showing that it satisfies all conditions in Definition 6.3.

2. $\Rightarrow$ Consequently, $A_1$ and $A_2$ are the only distinct, key-equal atoms of $db$. It follows that $db \setminus \{A_1\}$ and $db \setminus \{A_2\}$ are the two repairs of $db$. Both repairs satisfy $q$, because $\theta_2(q) \subseteq db \setminus \{A_1\}$ and $\theta_1(q) \subseteq db \setminus \{A_2\}$.

3. $\Rightarrow$ Obviously, $A_1, B_1 \in \theta_1(q) \subseteq db$.
4. $\Rightarrow$ Obviously, $A_2, B_2 \in \theta_2(q) \subseteq db$.

Assume that $\mu$ is a valuation such that $A_1 \in \mu(q) \subseteq db$. We show that $\mu(G) = B_1$. Let $\tau$ be a join tree for $q$. Let $F = H_0 \leadsto H_1 \leadsto H_2 \ldots \leadsto H_n = G$ be the path in $\tau$ 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 $F \rightarrow G$, we can assume $x \in L_{i+1} = \text{Vars}(H_i) \cap \text{Vars}(H_{i+1})$ such that $x \notin F^+, \theta_1(x) \neq \theta_2(x)$. 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 $\mu$ is a valuation such that $A_2 \in \mu(q) \subseteq db$, then $\mu(G) = B_2$.

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

Let $db_0$ be a witness database for $(F, G)$. We can assume valuations $\theta_1$ and $\theta_2$ 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 $db = \theta_1(q) \cup \theta_2(q)$ is a witness database for $(F, G)$. Let $q' = q \setminus \{F\}$ and $db' = db \setminus \{A_1, A_2\}$. Since $db'$ is consistent by the first item in Definition 6.3 and since $\theta_1(q'), \theta_2(q') \subseteq db'$, it follows by Lemma 4.3 that $\{\theta_1, \theta_2\} \models K(q')$. 

ACM Transactions on Database Systems, Vol. 0, No. 0, Article 0, Publication date: January 0000.
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 rewritings. Moreover, we show how such first-order rewriting can be constructed.

Assume towards a contradiction \( F \not\rightarrow G \). Let \( \tau \) be a join tree for \( q \). We can assume an edge \( e \) with label \( L \) on the unique path in \( \tau \) between \( F \) and \( G \) such that \( L \subseteq F^{+} \). Let \( \tau_{F} \) and \( \tau_{G} \) be the two join trees obtained from \( \tau \) by removing the edge \( e \), such that \( F \in \tau_{F} \) and \( G \in \tau_{G} \). This corresponds to the situation depicted in Fig. 12. Let \( \mu \) 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 \( \tau_{F} \) and \( \tau_{G} \), then, by the Connectedness Condition, \( x \) occurs in \( L \). Since \( A_{1} \) and \( A_{2} \) are key-equal, \( \theta_{1} \) and \( \theta_{2} \) agree on \( \mathcal{KVars}(F) \). From \( \{\theta_{1}, \theta_{2}\} = \mathcal{K}(q \setminus \{F\}) \) it follows that \( \theta_{1} \) and \( \theta_{2} \) agree on \( F^{+} \). Since \( L \subseteq F^{+} \), valuations \( \theta_{1} \) and \( \theta_{2} \) agree on each \( x \in L \). Thus, \( \mu \) is well-defined. We have \( \mu(F), \mu(G) \in \mu(q) \subseteq \mathbb{db} \) with \( \mu(F) = A_{1} \) and \( \mu(G) = B_{2} \), contradicting the fourth item in Definition 6.3. We conclude by contradiction \( F \not\rightarrow G \).

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 Fig. 3) or acyclic (see Fig. 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 \( F_{0} \leadsto F_{1} \leadsto F_{2} \ldots \leadsto F_{n-1} \leadsto F_{0} \) 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 \( \tau_{9} \) of query \( q_{9} \) shown in Fig. 10 (left). The attack graph of \( q_{9} \) is shown at the right. The cycles \( G \leadsto H \leadsto G \) and \( F \leadsto H \leadsto F \), both of size 2, are shortest. The cycle \( F \leadsto H \leadsto G \leadsto F \) 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 \( \text{CERTAINTY}(q) \) 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 rewritings. 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 non-Boolean 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 \( a_1, a_2, \ldots, a_n \),

\[
\text{db}_{\text{Sure}} \models q(a_1, a_2, \ldots, a_n) \iff \text{db} \models \varphi(a_1, a_2, \ldots, a_n).
\]

Definition 8.3 introduces our “rewrite function” \( \text{Rewrite}(F, q) \) 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, non-Boolean query \( q \setminus \{F\} \) whose free variables are \( \text{Vars}(F) \). 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 \( \varphi_1 \) 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 \( \{R(\vec{x}, y, y, a)\} \) 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 \( R(b, c_1, c_2, c_3, c_4) \) 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 \( \varphi \):

\[
\varphi = \exists x \exists z_1 \exists z_2 \exists z_3 \exists z_4 \left( R(\vec{x}, z_1, z_2, z_3, z_4) \land \forall z_1 \forall z_2 \forall z_3 \forall z_4 \left( R(\vec{x}, z_1, z_2, z_3, z_4) \rightarrow (z_1 = x \land z_3 = z_2 \land z_4 = a) \right) \right).
\]

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

\[
\varphi = \exists x \exists y \left( R(\vec{x}, x, y, y, a) \land \forall z_1 \forall z_2 \forall z_3 \forall z_4 \left( R(\vec{x}, z_1, z_2, z_3, z_4) \rightarrow (z_1 = x \land z_3 = z_2 \land z_4 = a) \right) \right).
\]

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

\[
\varphi = \exists x \exists y \left( R(\vec{x}, x, y, y, a) \land \forall z_1 \forall y \forall z_3 \forall z_4 \left( R(\vec{x}, z_1, y, z_3, z_4) \rightarrow (z_1 = x \land z_3 = y \land z_4 = a) \right) \right).
\]

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

— there exists a (unique) valuation \( \theta \) over \( \text{vars}(\vec{x}) \) such that \( \theta(\vec{x}) = \vec{b} \); and
— for each \( R \)-atom \( R(\vec{y}, c_1, c_2, \ldots, c_n) \) in \( \text{db} \), we can extend \( \theta \) to a valuation \( \theta^+ \) over \( \text{Vars}(F) \) such that \( \theta^+(y_i) = c_i \) for \( 1 \leq i \leq n \) and such that \( \text{db} = \text{true} \theta^+(q') \), 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 \( \theta \) can be extended from \( \text{vars}(\vec{x}) \) to \( \text{Vars}(F) \), we sequentially inspect \( \theta^+(y_1), \theta^+(y_2), \theta^+(y_3), \ldots \) in that order. Assume that the value of \( \theta^+ \) 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 \( y_i \in \text{vars}(\vec{x}) \), then \( \theta \) maps \( y_i \) to a constant. Then \( \theta^+ \) does not exist unless \( c_i = \theta(y_i) \).
   
   — If \( y_i \) is a variable that occurs in \( \langle y_1, \ldots, y_{i-1} \rangle \) 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 \( z_3 = y \).

Concerning the latter item, to determine whether valuation \( \theta \) can be extended from \( \text{vars}(\vec{x}) \) to \( \text{Vars}(F) \), we sequentially inspect \( \theta^+(y_1), \theta^+(y_2), \theta^+(y_3), \ldots \) in that order. Assume that the value of \( \theta^+ \) 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 \( y_i \in \text{vars}(\vec{x}) \), then \( \theta \) maps \( y_i \) to a constant. Then \( \theta^+ \) does not exist unless \( c_i = \theta(y_i) \).
   
   — If \( y_i \) is a variable that occurs in \( \langle y_1, \ldots, y_{i-1} \rangle \) 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 \( z_3 = y \).

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 \( R(\vec{x}, \vec{y}) \) be an atom of \( q \), and let \( \vec{y} = \langle y_1, y_2, \ldots, y_n \rangle \). Notice that \( \vec{y} \) can contain constants and repeated variables. Let \( \vec{z} = \langle z_1, z_2, \ldots, z_n \rangle \) 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 \( \langle \vec{x}, y_1, y_2, \ldots, y_{i-1} \rangle \), then \( z_i \) is identical to \( y_i \); (1)

2. otherwise \( z_i \) is a new variable and \( C \) contains \( z_i \neq y_i \). (2)

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

\[
\text{Rewrite}(R(\vec{x}, \vec{y}), q) = \exists \vec{v} \left( R(\vec{x}, \vec{y}) \land \forall \vec{z} \left( R(\vec{x}, \vec{z}) \rightarrow (C \land \varphi(\vec{v})) \right) \right).
\]

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

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

²In front of Theorem 8.13, it will be argued that the same rewrite function applies to nonBoolean queries by treating free variables as constants.
Let \( \varphi(x, y) \) be a certain first-order rewriting for \( q'(x, y) \), where \( q'(x, y) \) is the non-Boolean conjunctive query whose set of atoms is \( q \setminus \{ F \} \). Then,

\[
\text{Rewrite}(F, q) = \exists x \exists y \left( R(\vec{x}, x, y, a) \land \right.
\]
\[
\left. \forall z_1 \forall y \forall z_2 \forall z_4 \left( R(\vec{x}, z_1, y, z_3, z_4) \rightarrow (z_1 = x \land z_3 = y \land z_4 = a \land \varphi(x, y)) \right) \right).
\]

Our rewrite function \( \text{Rewrite}(F, q) \) starts with an existential quantification over the variables in \( \text{KVars}(F) \), 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 reifiable if for every database \( \text{db} \), \( \text{db} \models q \) implies \( \text{db} \models \text{Rewrite}(F, q) \).

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

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

Assume \( \text{db} \models q \). Since \( F \) is reifiable, we can assume a valuation \( \theta \) over \( \text{vars}(\vec{x}) \) such that \( \text{db} \models \text{Rewrite}(\theta, q) \). Then, for some valuation \( \mu \) over \( \text{vars}(\vec{y}) \setminus \text{vars}(\vec{x}) \), \( R(\theta(\vec{x}), \mu \circ \theta(\vec{y})) \in \text{db} \) and for every valuation \( \zeta \) over \( \text{vars}(\vec{z}) \), if \( R(\theta(\vec{x}), \zeta(\vec{z})) \in \text{db} \), then there exists a valuation \( \omega \) over \( \text{vars}(\vec{y}) \setminus \text{vars}(\vec{x}) \) such that \( \omega \circ \theta(\vec{y}) = \zeta(\vec{z}) \) and \( \text{db} \models \omega \circ \theta(\vec{y}) \).

Notice that, since \( \vec{z} \) is a sequence of distinct variables, every atom \( R(\theta(\vec{x}), \vec{b}) \) can be written as \( R(\theta(\vec{x}), \zeta(\vec{z})) \) 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 \( \text{db} \models \zeta \circ \theta(q') \).

**Proof.** By Definition 8.3, \( \text{vars}(\vec{y}) = \text{vars}(\vec{x}) \cup \text{vars}(\vec{z}) \) and \( \text{vars}(\vec{x}) \subseteq \text{vars}(\vec{z}) \). Therefore, \( \text{vars}(\vec{z}) \subseteq \text{vars}(\vec{x}) \cup \text{vars}(\vec{x}) \cup \text{vars}(\vec{z}) \). Since \( \text{vars}(\vec{z}) \subseteq \text{vars}(\vec{x}) \cup \text{vars}(\vec{z}) \), 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 \( \text{vars}(\vec{x}) \cup \text{vars}(\vec{z}) \). 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, \( y_i \) 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:

\[
\zeta(z_i) = \zeta \circ \theta(y_i).
\]

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

\[
\omega \circ \theta(y_i) = \zeta(z_i).
\]

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 \( \langle y_1, y_2, \ldots, y_n \rangle \) 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 $\omega \circ \theta(g) = \zeta(z)$ by (1), it follows $\omega \circ \theta(y_i) = \zeta(y_i)$. 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 $\omega(y_i) = \zeta(y_i)$.

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

We next show $\theta(q)$. Since $\theta(q')$ by (1), it suffices to show that for every $y \in \text{vars}(g) \setminus \text{vars}(x)$, $\omega(y) = \zeta(y)$. Assume $i \geq 1$ such that $y_i$ is a variable that does not occur in $x$. We need to show $\omega(y_i) = \zeta(y_i)$. Two cases occur: if $y_i$ occurs in ${y_1, y_2, \ldots, y_{i-1}}$, then the desired result follows from the third case above; if $y_i$ does not occur in ${y_1, y_2, \ldots, y_{i-1}}$, 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 $\theta$ over $\text{vars}(x)$,
- for some valuation $\mu$ over $\text{vars}(g) \setminus \text{vars}(x)$,
- $R(\theta(z), \mu \circ \theta(g)) \in \text{db}$ and
- for every valuation $\zeta$ over $\text{vars}(z)$, if $R(\theta(z), \zeta(z)) \in \text{db}$,
  then $\zeta \circ \theta(C)$ and $\text{db} \models_c \zeta \circ \theta(q')$.

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

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 $\text{db}$, for all repairs $r$ and $s$ of $\text{db}$, there exists a repair $\text{rep}$ of $r \cup s$ (and hence of $\text{db}$) such that for every valuation $\theta$ over $\text{KVars}(F)$, $\text{rep} \models \theta(q)$ implies $r \models \theta(q)$ and $s \models \theta(q)$. The construction of such repair $\text{rep}$ 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 $\text{rep}$ be a repair of some database. For an atom $F$ of $q$, we define:

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

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

$$[r, s] = [r_0, s_0], [r_1, s_1], \ldots, [r_n, s_n]$$

where for each $i \in \{0, 1, \ldots, n-1\}$, there exist atoms $A \in r_i$ and $B \in s_i$ 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 $r_i$ and $r_{i+1} = (r_i \setminus \{A\}) \cup \{B\}$ and $s_{i+1} = s_i$; or
- $B$ is relevant for $q$ in $s_i$ and $r_{i+1} = r_i$ and $s_{i+1} = (s_i \setminus \{B\}) \cup \{A\}$.

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

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

$$r_0 = \{R(a, b), S(a, b), R(c, 1), S(c, 2), R(c, f), S(c, f)\},$$
$$s_0 = \{R(a, 1), S(a, 1), R(c, d), S(c, d), R(c, f), S(c, f)\}.$$
Since \( R(q, b) \) is relevant for \( q_3 \) in \( r_0 \), it is replaced with \( R(q, 1) \), giving:

\[
\begin{align*}
    r_1 &= \{ R(q, 1), S(q, b), R(q, 1), S(q, 2), R(q, f), S(q, f) \}, \\
    s_1 &= \{ R(q, 1), S(q, 1), R(q, d), S(q, d), R(q, f), S(q, f) \}.
\end{align*}
\]

Since \( S(q, 1) \) is relevant for \( q_3 \) in \( s_1 \), it is replaced with \( S(q, b) \), giving:

\[
\begin{align*}
    r_2 &= \{ R(q, 1), S(q, b), R(q, 1), S(q, 2), R(q, f), S(q, f) \}, \\
    s_2 &= \{ R(q, 1), S(q, b), R(q, d), S(q, d), R(q, f), S(q, f) \}.
\end{align*}
\]

Since \( R(q, d) \) is relevant for \( q_3 \) in \( s_2 \), it is replaced with \( R(q, 1) \), giving:

\[
\begin{align*}
    r_3 &= \{ R(q, 1), S(q, b), R(q, 1), S(q, 2), R(q, f), S(q, f) \}, \\
    s_3 &= \{ R(q, 1), S(q, b), R(q, 1), S(q, d), R(q, f), S(q, f) \}.
\end{align*}
\]

The uniformisation terminates, because the relevant atoms \( R(q, f) \) and \( S(q, f) \) are the same in \( r_3 \) and \( s_3 \). It is easy to see that every uniformisation must eventually terminate because the cardinality of \( r \cup s \) 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 \( G \in q \setminus \{ F \} \), \( G \not\leftarrow F \). Let \( r, s \) be two repairs of the same database \( db \). Let \( [r_n, s_n] \) be the last element in a uniformisation of \([r, s]\). Then,

1. \( r_n \) is a repair of \( db \); and
2. \( \text{Reify}(q, F, r_n) \subseteq \text{Reify}(q, F, r) \cap \text{Reify}(q, F, s) \).

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

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

1. \( \text{Reify}(q, F, \text{rep}) \neq \{ \} \). Then, we can assume a valuation \( \theta \) over \( K\text{Vars}(F) \) such that for each repair \( \text{rep}' \) of \( db \), \( \text{rep}' \models \theta(q) \). It follows \( db|_{\text{sure}} \models \theta(q) \).
2. \( \text{Reify}(q, F, \text{rep}) = \{ \} \). Then, for every valuation \( \theta \) over \( K\text{Vars}(F) \), \( \text{rep} \not\models \theta(q) \). It follows \( \text{rep} \not\models q \). Since \( \text{rep} \) is a repair of \( db \), \( db|_{\text{sure}} \models q \).

Since \( db \) is arbitrary, it follows that \( F \) is reifiable in \( q \). \( \square \)

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 incoming edge, it follows that an acyclic attack graph contains a non-attacked atom, which is reifiable by Corollary 8.11 and thus allows the application of Lemma 8.6. Second, constructing certain first-order rewritings for non-Boolean 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 \( \varphi \) 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 \( \varphi \) 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 $q_3 = \{R(x, y), S(x, y)\}$, the atom $R(x, y)$ is not attacked. A certain first-order rewriting for $q_3$ is shown next.

$$\text{Rewrite}(R(x, y), q_3) = \exists x \forall y (R(x, y) \land \varphi(x, y))$$

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:

$$\varphi(x, y) = S(x, y) \land z \{z \rightarrow \varphi(x, y)\}$$

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

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

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

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

1. $\text{CERTAINTY}(q)$ 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 $\text{CERTAINTY}(q)$ 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 $\text{cycle}(u) = \{(G_1, G_2), (G_2, G_3), \ldots, (G_{\ell-1}, G_\ell), (G_\ell, G_1)\}$ where $G_1 \prec G_2 \prec \cdots \prec G_\ell$ are all the atoms of $q$ in which $u$ occurs, i.e., $\{G \in q \mid u \in \text{Vars}(G)\} = \{G_1, \ldots, G_\ell\}$. From a graph perspective, $\text{cycle}(u)$ is a directed cycle graph through all atoms of $q$ that contain $u$.

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

— the vertices of $\text{Witness}(F)$ are the atoms of $q$; and
— the set of edges of $\text{Witness}(F)$ is $\bigcup \{\text{cycle}(u) \mid u \in U \setminus F^{+\sim}q\}$.
Assume there is a directed path from $F$ to $G$, where $F$ and $G$ are acyclic conjunctive queries without self-join. Let $U$ be the set of variables that occur in $F$. By Corollary 8.14 and Lemma 7.3, given an acyclic conjunctive query $q$ without self-join, 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 $q$.

Example 9.3. It can be verified that $q_{10} = \{R_1(x, y, u), R_2(u, y), R_3(z, v, v, y), R_4(z, v, y, w)\}$ is acyclic. Let $F = R_1(x, y, u)$ 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 Fig. 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 Fig. 11 is associated to the occurrence of the variable $w$ in $R_4(z, v, y, w)$. 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 Witness$(F)$.

**Proof.** Let $U$ be the set of variables that occur in $q$. Let $\tau$ be a join tree for $q$.

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

2. $2 \Rightarrow 1$ Assume there is a directed path $(G_0, G_1, \ldots, G_n)$ in 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 Witness$(F)$, we can assume a variable $x_i \in U \setminus F^{+q}$ such that $(G_{i-1}, G_i) \in \text{cycle}(x_i)$. Since $x_i \in \text{Vars}(G_{i-1}) \cap \text{Vars}(G_i)$, by the Connectedness Condition for join trees, $x_i$ occurs in every atom on the unique path in $\tau$ 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 $F \Rightarrow G_j$ for all $j \in \{1, \ldots, n\}$. It follows $F \Rightarrow G$. □

9.2. Quadratic Time Algorithm

By Corollary 8.14 and Lemma 7.3, given an acyclic conjunctive query $q$ without self-join, 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 $q$. 

ACM Transactions on Database Systems, Vol. 0, No. 0, Article 0, Publication date: January 0000.
**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 \( \text{ATTACK} \). The Boolean variable \( \text{FO} \) indicates whether \( \text{CERTAINTY}(q) \) 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, \ldots, \( n \).
- Variables are encoded by integers 1, 2, \ldots, \( m \).
- \( \text{KEY}[1:n] \) is an array of lists of variables such that for \( 1 \leq F \leq n \), entry \( \text{KEY}[F] \) contains all variables that occur in the primary key of atom \( F \).
- \( \text{NONKEY}[1:n] \) is an array of lists of variables such that for \( 1 \leq F \leq n \), entry \( \text{NONKEY}[F] \) 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.

- \( \text{KEYCOUNT}[1:n] \) is an array of integers such that for \( 1 \leq F \leq n \), entry \( \text{KEYCOUNT}[F] \) is the number of distinct variables in the primary key of atom \( F \). \( \text{KEYCOUNT} \) remains unchanged after initialization.
- \( \text{COUNT}[1:n] \) is an array of integers.
- \( \text{KEYCO}[1:m] \) is an array of lists of atoms such that for \( 1 \leq u \leq m \), entry \( \text{KEYCO}[u] \) contains all atoms in which variable \( u \) occurs at some primary key position. \( \text{KEYCO} \) remains unchanged after initialization.
- \( \text{CO}[1:m] \) is an array of lists of atoms such that for \( 1 \leq u \leq m \), entry \( \text{CO}[u] \) contains the atoms in which variable \( u \) occurs. \( \text{CO} \) remains unchanged after initialization.
- \( \text{WITNESSF}[1:n] \) is an adjacency list representation of a directed graph whose vertices are the atoms of \( q \). For \( 1 \leq G \leq n \), entry \( \text{WITNESSF}[G] \) 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 \( F \).
- \( \text{ATTACK}[1:n,1:n] \) is an adjacency matrix representing a directed graph whose vertices are the atoms of \( q \). Entry \( \text{ATTACK}[F,G] \) 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 \( q \).

### Algorithm:

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

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, \( n \); \( \text{KEY} \) and \( \text{NONKEY} \) are arrays of lists that store for each atom \( F \) the variables of \( \text{KVars}(F) \) and \( \text{Vars}(F) \setminus \text{KVars}(F) \) respectively. Hereinafter, the length of this encoding will be denoted by \( \text{len}(q) \). Notice that \( \text{len}(q) \) 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 \( q_{10} = \{ R_1(x, y, u), R_2(u, y), R_3(z, v, y), R_4(z, v, y, w) \} \), 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 fol-
Procedure Initialization

\[
\begin{align*}
\text{foreach } u \leftarrow 1 \text{ to } m \text{ do} & \\
& \text{KEYCO}[u] \leftarrow \emptyset; \\
& \text{CO}[u] \leftarrow \emptyset; \\
\end{align*}
\]

\[
\begin{align*}
\text{end} \\
\text{foreach } F \leftarrow 1 \text{ to } n \text{ do} & \\
& i \leftarrow 0; \\
& \text{foreach } u \in \text{KEY}[F] \text{ do} & \\
& & \text{add } F \text{ to KEYCO}[u]; \\
& & i \leftarrow i + 1; \\
& \text{end} \\
& \text{KEYCOUNT}[F] \leftarrow i; \\
& \text{foreach } u \in \text{KEY}[F] \cup \text{NONKEY}[F] \text{ do} & \\
& & \text{add } F \text{ to CO}[u]; \\
& \text{end} \\
& \text{foreach } G \leftarrow 1 \text{ to } n \text{ do} & \\
& & \text{ATTACK}[F, G] \leftarrow 0; \\
& \text{end} \\
\text{end} \\
\text{FO} \leftarrow \text{true};
\end{align*}
\]

Following representation for \( q \) (for readability, references to atoms are typeset in bold):

\[
\begin{align*}
\text{KEY}[1] &= (4, 5) & \text{NONKEY}[1] &= (1) \\
\text{KEY}[2] &= (1) & \text{NONKEY}[2] &= (5) \\
\text{KEY}[3] &= (2, 6) & \text{NONKEY}[3] &= (5) \\
\text{KEY}[4] &= (6) & \text{NONKEY}[4] &= (2, 3, 5)
\end{align*}
\]

Note incidentally that there is no need to know that \( v \) also occurs at a non-primary key position in the third atom.

The Initialization procedure fills in the following two data structures, obviously in \( O(\text{len}(q)) \) time: an array \( \text{CO} \) of lists that stores for each variable \( u \) the atoms in which \( u \) occurs; an array \( \text{KEYCO} \) 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):

\[
\begin{align*}
\text{CO}[1] &= (1, 2) & \text{KEYCO}[1] &= (2) \\
\text{CO}[2] &= (3, 4) & \text{KEYCO}[2] &= (3) \\
\text{CO}[3] &= (4) & \text{KEYCO}[3] &= () \\
\text{CO}[4] &= (1) & \text{KEYCO}[4] &= (1) \\
\text{CO}[5] &= (1, 2, 3, 4) & \text{KEYCO}[5] &= (1) \\
\text{CO}[6] &= (3, 4) & \text{KEYCO}[6] &= (3, 4)
\end{align*}
\]

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

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 \( \text{WITNESSF} \) using an adjacency list representation [Dasgupta et al. 2008, page 82].
**Procedure** BuildAttackGraph

```plaintext
foreach F ← 1 to n do
    FPLUS ← KEY[F]; // variable FPLUS is used to compute F+q
    foreach G ← 1 to n do
        WITNESSF[G] ← ∅;
        COUNT[G] ← KEYCOUNT[G];
        if KEYCOUNT[G] = 0 and G ≠ F then
            FPLUS ← FPLUS ∪ NONKEY[G];
    end
    COUNT[F] ← m + 1; // high value that will never be decremented to 0
    UPDATE ← FPLUS;
    while UPDATE ≠ ∅ 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, ..., m} \ FPLUS do // construct the witness graph of F
        Encode the edges of cycle(u) in WITNESSF, as follows. If G₁, G₂, ..., G_ℓ are the atoms in CO[u] in increasing order, then add G₁ to WITNESSF[G_ℓ], and for 1 ≤ i < ℓ, add Gᵢ to WITNESSF[Gᵢ+1].
    end
    foreach atom G ≠ 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
```

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

```
WITNESS[1] = (2)
WITNESS[2] = (1)
WITNESS[3] = (4)
WITNESS[4] = (3, 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 O(len(q)), once its adjacency list representation WITNESS
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 $\text{ATTACK}$. 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 $\text{LINCLOSURE}$ algorithm [Beeri and Bernstein 1979, Algorithm 2] [Maier 1983, page 66]. The set $F^{+q}$ is constructed in the variable $\text{FPLUS}$, which is initialized to $\text{KEY}[F]$. Then, for every atom $G \neq F$ with $\text{KVars}(G) = \{\}$, all variables of $\text{NONKEY}(G)$ are added to $\text{FPLUS}$. The further computation relies on a counter $\text{COUNT}(G)$ for each atom $G$, which is initialized to the cardinality of $\text{KVars}(G)$. The computation of $F^{+q}$ loops exactly once through each variable $u$ that has been or will be inserted into $\text{FPLUS}$, and decrements $\text{COUNT}(G)$ for every atom $G$ in $\text{KEYCO}[u]$. If $\text{COUNT}(G)$ reaches 0 for some $G \neq F$, then we know that all variables of $\text{KVars}(G)$ belong to $F^{+q}$, so we add each variable of $\text{NONKEY}(G)$ to $\text{FPLUS}$ (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 $\text{LINCLOSURE}$ [Beeri and Bernstein 1979; Maier 1983], it can be shown that $F^{+q}$ is correctly computed in the variable $\text{FPLUS}$ in $O(|\text{len}(q)|)$ time.

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

10. CONCLUDING REMARKS

The complexity of the problem $\text{CERTAINTY}(q)$ for conjunctive queries $q$ has received a growing research interest in recent years. It is a special case of consistent query 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 $\text{CERTAINTY}(q)$ when $q$ ranges over the class of acyclic conjunctive queries without self-join. This result is of practical interest, because if $\text{CERTAINTY}(q)$ is first-order expressible, then $\text{CERTAINTY}(q)$ 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:

— 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 $\text{CERTAINTY}(q)$ when $q$ is cyclic.

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

APPENDIX

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 Fig. 12: the join tree $\tau$ consists of two join trees, $\tau_F$ and $\tau_G$, linked by an edge with label $L$; the atom $F$ occurs in $\tau_F$, and $G$ in $\tau_G$. 

ACM Transactions on Database Systems, Vol. 0, No. 0, Article 0, Publication date: January 0000.
LEMMA A.1. Let \( q \) be a Boolean conjunctive query. Let \( \tau \) 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 \( \tau_F \) and \( \tau_G \) be the join trees obtained from \( \tau \) 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 \( \tau_G \). For every \( q' \subseteq q \), if \( \mathcal{K}(q') \models \text{KVars}(F) \rightarrow S \), 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 \( \{ x \in U \mid \mathcal{K}(q') \models \text{KVars}(F) \rightarrow x \} \) 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):

\[
\text{KVars}(F) = S_0 \quad H_1 \\
S_1 \quad H_2 \\
\vdots \quad \vdots \\
S_{k-1} \quad H_k \\
S_k 
\]

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 \( H_i \in q' \) such that \( \text{KVars}(H_i) \subseteq S_{i-1} \) and \( S_i = S_{i-1} \cup \text{Vars}(H_i) \).

Then, for every \( x \in U \), \( \mathcal{K}(q') \models \text{KVars}(F) \rightarrow x \) 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 \( \text{KVars}(F) \cap U_G \subseteq L \) 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 \( \tau_F \). By the Connectedness Condition, \( \text{Vars}(H_{i+1}) \cap U_G \subseteq L \). 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 \( \tau_G \), hence \( \text{Vars}(H_{i+1}) \subseteq U_G \). We have \( S_{i+1} \cap U_G = (S_i \cap U_G) \cup \text{Vars}(H_{i+1}) \). Since \( \mathcal{K}(q') \models L \rightarrow S_i \cap U_G \) by the induction hypothesis, it suffices to show \( \mathcal{K}(q') \models L \rightarrow \text{Vars}(H_{i+1}) \).

Since \( \text{KVars}(H_{i+1}) \subseteq S_i \cap U_G \) and \( \mathcal{K}(q') \models L \rightarrow S_i \cap U_G \) by the induction hypothesis, we have \( \mathcal{K}(q') \models L \rightarrow \text{KVars}(H_{i+1}) \). Since \( H_{i+1} \in q' \), the set \( \mathcal{K}(q') \) contains \( \text{KVars}(H_{i+1}) \rightarrow \text{Vars}(H_{i+1}) \). Consequently, \( \mathcal{K}(q') \models L \rightarrow \text{Vars}(H_{i+1}) \).

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

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

PROOF. Assume \( F \leadsto G \) and \( G \leadsto H \). Let \( \tau \) be a join tree for \( q \). We distinguish three cases:
Case $H \in [F,G]_{\tau}$. By item (2) in Lemma 4.9, $F \Rightarrow_{\tau} H$.

Case $F \in [G,H]_{\tau}$. By item (2) in Lemma 4.9, $G \Rightarrow_{\tau} F$.

Case $H \not\in [F,G]_{\tau}$ and $F \not\in [G,H]_{\tau}$. Thus, for some $J \not\in \{F,H\}$, the join tree $\tau$ 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 $F \not\Rightarrow_{\tau} H$ and $G \not\Rightarrow_{\tau} F$.

Since $F \Rightarrow_{\tau} G$ and $F \Rightarrow_{\tau} H$, we can assume an edge $e_0$ with label $L_0$ on the path between $J$ and $H$ such that $K(q \setminus \{F\}) \models \text{KVars}(F) \rightarrow L_0$. From item (3) in Lemma 4.9, it follows $K(q \setminus \{F,G\}) \models \text{KVars}(F) \rightarrow L_0$, hence

$$K(q \setminus \{G\}) \models \text{KVars}(F) \rightarrow L_0.$$  \hspace{1cm} (4)

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

$$K(q \setminus \{G\}) \models \text{KVars}(G) \rightarrow L_1.$$  \hspace{1cm} (5)

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

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

Thus, the desired result holds in all cases. \qed

The proof of Lemma 7.3 is given next.

**Proof of Lemma 7.3.** Assume that $F_0 \leadsto F_1 \leadsto F_2 \ldots \leadsto F_{n-1} \leadsto F_0$ 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 $F_1 \Rightarrow F_0$. Then $F_0 \Rightarrow F_1 \leadsto F_0$ is a cycle of size 2, contradicting that some shortest cycle has size at least 3.

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

We conclude $n = 2$ by contradiction. \qed

**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 \( G \in q \setminus \{ F \} \), \( G \not\supseteq F \). Let \( \text{rep} \) be a repair of some database. Let \( A \in \text{rep} \) such that \( A \) is relevant for \( q \) in \( \text{rep} \). Let \( B \) be key-equal to \( A \) and \( \text{rep}_B = (\text{rep} \setminus \{ A \}) \cup \{ B \} \). Then, \( \text{Reify}(q, F, \text{rep}_B) \subseteq \text{Reify}(q, F, \text{rep}) \).

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 \( \zeta \) be a valuation over \( \text{KVars}(F) \) such that \( \text{rep}_B \models \zeta(q) \). Let \( U \) be the set of variables that occur in \( q \). We can assume a valuation \( \zeta^+ \) over \( U \) such that \( \zeta^+ [\text{KVars}(F)] = \zeta [\text{KVars}(F)] \) and \( \zeta^+(q) \subseteq \text{rep}_B \). Thus, \( \zeta^+ \) extends \( \zeta \) to \( U \). We need to show \( \text{rep} \models \zeta(q) \), which is obvious if \( B \not\subseteq \zeta^+(q) \). Assume next \( B \in \zeta^+(q) \). Since \( A \) is relevant for \( q \) in \( \text{rep} \), we can assume a valuation \( \mu \) over \( U \) such that \( A \in \mu(q) \subseteq \text{rep} \). We can assume \( G \in q \) such that \( A, B \) have the same relation name as \( G \). Thus, \( G \not\supseteq F, A = \mu(G) \), and \( B = \zeta^+(G) \). Let \( q' = q \setminus \{ G \} \). Let \( \text{rep}' = \text{rep}_B \setminus \{ B \} = \text{rep}_B \setminus \{ A \} \). Since \( q' \) contains no atom with the same relation name as \( G \) (no self-join), \( \zeta^+(q') \subseteq \text{rep}' \) and \( \mu(q') \subseteq \text{rep}' \). Moreover, \( \zeta^+ [\text{KVars}(G)] = \mu [\text{KVars}(G)] \), because \( A \) and \( B \) are key-equal. Let \( \tau \) be a join tree for \( q \). Since \( G \not\supseteq F \), we can assume an edge \( e \) with label \( L \) on the unique path in \( \tau \) between \( G \) and \( F \) such that \( L \subseteq G^+ \). Since \( \text{K}(q \setminus \{ G \}) = \text{K}(q') \), we have \( \text{K}(q') \models \text{KVars}(G) \rightarrow L \).

By Lemma 4.3, \( \zeta^+(L) = \mu(L) \). Let \( \tau_G, \tau_F \) be the two join trees obtained from \( \tau \) by cutting the edge \( e \) with label \( L \), where \( \tau_G \) contains \( G \), and \( \tau_F \) contains \( F \). This corresponds to the situation depicted in Fig. 12. Let \( \kappa \) 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 \( \tau_G \) and \( \tau_F \), then, by the Connectedness Condition, \( x \) occurs in \( L \), hence \( \mu(x) = \zeta^+(x) \). Obviously, \( \kappa(q) \subseteq \text{rep} \). It follows \( \text{rep} \models \zeta(q) \). \( \square \)

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 uniformisation, we obtain:

\[
\text{Reify}(q, F, r_n) \subseteq \text{Reify}(q, F, r), \quad \text{and} \quad \text{Reify}(q, F, s_n) \subseteq \text{Reify}(q, F, s) .
\]

We show \( \text{Reify}(q, F, r_n) \subseteq \text{Reify}(q, F, s_n) \). Let \( \theta \) be a valuation over \( \text{KVars}(F) \) such that \( r_n = \theta(q) \). Let \( U \) be the set of variables that occur in \( q \). We can assume a valuation \( \theta^+ \) that extends \( \theta \) to \( U \) such that \( \theta^+(q) \subseteq r_n \). Then, every atom in \( \theta^+(q) \) is relevant for \( q \) in \( r_n \). By our uniformisation construction, \( \theta^+(q) \subseteq s_n \). It follows \( s_n = \theta(q) \). Consequently, \( \text{Reify}(q, F, r_n) \subseteq \text{Reify}(q, F, s_n) \), hence \( \text{Reify}(q, F, r_n) \subseteq \text{Reify}(q, F, s) \) by (7). \( \square \)

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

ACM Transactions on Database Systems, Vol. 0, No. 0, Article 0, Publication date: January 0000.
Let $\theta$ 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 $F^{\tau+q} \setminus \{ x \} \not\subseteq (F')^{\tau+q}$. The computation of the attribute closure $F^{\tau+q}$ 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):

$$\text{KVars}(F) = S_0 \quad H_1
S_1 \quad H_2
\vdots \quad \vdots
S_{k-1} \quad H_k
S_k
$$

where

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

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

$$\text{KVars}(F') = S_0 \setminus \{ x \} \quad H'_1
S_1 \setminus \{ x \} \quad H'_2
\vdots \quad \vdots
S_{k-1} \setminus \{ x \} \quad H'_k
S_k \setminus \{ x \}$$

Obviously, for every $i \in \{1, 2, \ldots, k\}$, we have:
- $H'_i \in q' \setminus \{ F' \}$,
- KVars($H'_i$) $\subseteq$ $S_{i-1} \setminus \{ x \}$, and
- $S_i \setminus \{ x \} = (S_{i-1} \setminus \{ x \}) \cup$ Vars($H'_i$).

This corresponding sequence shows $F^{\tau+q} \setminus \{ x \} \not\subseteq (F')^{\tau+q}$.

Let $F, G$ be distinct atoms of $q$ such that $F \not\rightarrow L G$. Then, for some edge $H \rightarrow L J$ on the path between $F$ and $G$ in $\tau$, we have $L \subseteq F^{\tau+q}$. Obviously, the path between $F'$ and $G'$ in $\tau'$ contains an edge $H' \rightarrow L'$ with $L' = L \setminus \{ x \}$. From $L \setminus \{ x \} \subseteq F^{\tau+q} \setminus \{ x \}$ and $F^{\tau+q} \setminus \{ x \} \not\subseteq (F')^{\tau+q}$, it follows $L' \not\subseteq (F')^{\tau+q}$. Consequently, $F' \not\rightarrow L' G'$. It follows that the attack graph of $\tau'$ 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 $\tau$ be a join tree for $q$. Let $\tau_F$ be the directed rooted join tree obtained from $\tau$ by selecting $F$ as the root. Let $F_1, F_2, \ldots, F_k$ be the children of $F$ in $\tau_F$. For $i \in \{1, 2, \ldots, k\}$, let $\tau_i$ be the vertex-induced subtree of $\tau_F$ induced by $F_i$ and all descendants of $F_i$. The situation is depicted in Fig. 13. Clearly, if $e$ is an edge adjacent to $F$, then the label of $e$ is the empty set. Let $\tau'$ be the tree that contains $\tau_1, \tau_2, \ldots, \tau_k$ and additional edges $F_1 \rightarrow F_2, F_2 \rightarrow F_3, \ldots, F_{k-1} \rightarrow F_k$. It is straightforward that $\tau'$ is a join tree for $q \setminus \{ F \}$ and that the attack graph of $\tau'$ 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 \( R_1(\vec{x}_1, \vec{y}_1) \), \( R_2(\vec{x}_2, \vec{y}_2) \), \ldots, \( R_n(\vec{x}_n, \vec{y}_n) \) be a topological sorting of the atoms of \( q \) with respect to the attack graph. \( R_1(\vec{x}_1, \vec{y}_1) \) 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 \( \text{vars}(\vec{x}_1) \cup \text{vars}(\vec{y}_1) \). Let \( q'(\vec{v}) \) be the nonBoolean conjunctive query whose set of atoms is \( q \setminus \{ R_1(\vec{x}_1, \vec{y}_1) \} \). Let \( \vec{c} = \langle c_1, c_2, \ldots, c_n \rangle \) be a sequence of new constants. Let \( q'_c \) be the query obtained from \( q' \) by replacing each \( v_i \) with \( c_i \). It follows from Lemmas C.1 and C.2 that \( q'_c \) is an acyclic conjunctive query with an acyclic attack graph. By the induction hypothesis, \( q'_c \) has a certain first-order rewriting \( \varphi \). Let \( \varphi(\vec{v}) \) be the formula obtained from \( \varphi \) 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. \( \square \)

Acknowledgments
Theorem 6.1 was brought up 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


ACM Transactions on Database Systems, Vol. 0, No. 0, Article 0, Publication date: January 0000.