Consistent Query Answering under Primary Key Constraints

Jef Wijsen

Université de Mons-Hainaut
soon: Université de Mons
Table of Contents

- Consistent First-order Rewriting
- The Class $C_{rooted}$
- Deciding $C_{rooted}$
Motivation

- How to deal with inconsistency in databases?
- Inconsistency entails incomplete database (i.e. set of possible databases, called repairs).

<table>
<thead>
<tr>
<th>EMP</th>
<th>Name</th>
<th>Status</th>
<th>Dept</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ed</td>
<td>married</td>
<td>Toys</td>
</tr>
<tr>
<td></td>
<td>Ed</td>
<td>single</td>
<td>Toys</td>
</tr>
</tbody>
</table>

PRIMARY KEY(Name)

First repair:
- Ed married Toys

Second repair:
- Ed single Toys

We are interested in determining whether a Boolean query $q$ is certain (i.e. whether $q$ is true in every repair).

$\text{EMP}(\text{Ed}; y; \text{Toys})$ is certain.

$\text{EMP}(\text{Ed}; \text{married}; z; \text{Toys})$ is possible but not certain.
Motivation

- How to deal with inconsistency in databases?
- Inconsistency entails incomplete database (i.e. set of possible databases, called repairs).

<table>
<thead>
<tr>
<th>EMP</th>
<th>Name</th>
<th>Status</th>
<th>Dept</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ed</td>
<td>married</td>
<td>Toys</td>
<td></td>
</tr>
<tr>
<td>Ed</td>
<td>single</td>
<td>Toys</td>
<td></td>
</tr>
</tbody>
</table>

PRIMARY KEY(Name)

First repair:

Ed married Toys

Second repair:

Ed single Toys

We are interested in determining whether a Boolean query $q$ is certain (i.e. whether $q$ is true in every repair).

- $\exists y \text{EMP}(\underline{Ed}, y, \underline{Toys})$ is certain.
- $\exists z \text{EMP}(\underline{Ed}, \text{married}, z)$ is possible but not certain.
Problem Statement

- Assume a database schema where primary keys are the only constraints. Primary keys will be underlined.
- A repair of a not-necessarily-consistently-consistent database $\text{db}$ is a maximal subset of $\text{db}$ that contains no two atoms $R(\overline{a}, \overline{b}), R(\overline{a}, \overline{c})$ with $\overline{b} \neq \overline{c}$. 
Problem Statement

- Assume a database schema where **primary keys** are the only constraints. Primary keys will be underlined.

- A **repair** of a not-necessarily-consistently database $db$ is a maximal subset of $db$ that contains no two atoms $R(\overline{a}, \overline{b}), R(\overline{a}, \overline{c})$ with $\overline{b} \neq \overline{c}$.

- Given a database schema $S$ and a Boolean query $q$ over $S$, let

$$CQA_S(q) := \{db \mid \text{every repair of } db \text{ satisfies } q\}$$
Problem Statement

- Assume a database schema where primary keys are the only constraints. Primary keys will be underlined.

- A repair of a not-necessarily-consistent database $\mathbf{db}$ is a maximal subset of $\mathbf{db}$ that contains no two atoms $R(\overline{a}, \overline{b}), R(\overline{a}, \overline{c})$ with $\overline{b} \neq \overline{c}$.

- Given a database schema $\mathcal{S}$ and a Boolean query $q$ over $\mathcal{S}$, let

  $$\text{CQA}_{\mathcal{S}}(q) := \{ \mathbf{db} \mid \text{every repair of } \mathbf{db} \text{ satisfies } q \}$$

- The schema $\mathcal{S}$ will be clear from the query. E.g.,

  $$\exists x \exists y \exists z \exists w (R(x, y, z) \land S(z, w))$$
Complexity

- We limit attention to conjunctive queries.
- In $\text{coNP}$.

If $db \not\in \text{CQA}(q)$, then there exists a repair $\text{rep}$ of $db$ such that $\text{rep}$ falsifies $q$.
Complexity

- We limit attention to conjunctive queries.
- In $\text{coNP}$.
  
  If $\text{db} \not\in \text{CQA}(q)$, then there exists a repair $\text{rep}$ of $\text{db}$ such that $\text{rep}$ falsifies $q$.

- $\text{CQA}(q_1)$ is $\text{coNP}$-complete for [CM05]:

$$q_1 = \exists x \exists y \exists z (R(x, z) \land S(y, z)).$$
We limit attention to conjunctive queries.

In $\text{coNP}$. If $\text{db} \not\models \text{CQA}(q)$, then there exists a repair $\text{rep}$ of $\text{db}$ such that $\text{rep}$ falsifies $q$.

$\text{CQA}(q_1)$ is $\text{coNP}$-complete for [CM05]:

$$q_1 = \exists x \exists y \exists z (R(x, z) \land S(y, z)).$$

$\text{CQA}(q_2)$ is $\text{coNP}$-complete for (see next slide):

$$q_2 = \exists x \exists y \exists z (C(x, z) \land C(y, z) \land E(x, y)).$$
Reduction From Graph 3-Colorability

\[ q_2 = \exists x \exists y \exists z (\text{Color}(x, z) \land \text{Color}(y, z) \land \text{Edge}(x, y)) \]

graph is 3-colorable \iff some repair falsifies \( q_2 \)

<table>
<thead>
<tr>
<th>Edge</th>
<th>From</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Color</th>
<th>Vertex</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>blue</td>
<td>1</td>
<td>yellow</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>red</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>blue</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>yellow</td>
</tr>
</tbody>
</table>
Reduction From Graph 3-Colorability

\[ q_2 = \exists x \exists y \exists z (\text{Color}(x, z) \land \text{Color}(y, z) \land \text{Edge}(x, y)) \]

graph is 3-colorable \iff some repair falsifies \( q_2 \)

<table>
<thead>
<tr>
<th>Edge</th>
<th>From</th>
<th>To</th>
<th>Color</th>
<th>Vertex</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td></td>
<td>1</td>
<td>blue</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>3</td>
<td></td>
<td>2</td>
<td>red</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>3</td>
<td></td>
<td>3</td>
<td>yellow</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>4</td>
<td></td>
<td>4</td>
<td>blue</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
First-order Definability

We say that $\text{CQA}(q)$ is first-order definable if there exists a first-order sentence $\psi$ such that for every database $db$:

$$db \in \text{CQA}(q) \iff db \text{ satisfies } \psi$$
First-order Definability

- We say that $\text{CQA}(q)$ is first-order definable if there exists a first-order sentence $\psi$ such that for every database $db$:

$$db \in \text{CQA}(q) \iff db \text{ satisfies } \psi$$

- $\psi$, if it exists, is called a consistent first-order rewriting for $q$. 
We say that \( \text{CQA}(q) \) is first-order definable if there exists a first-order sentence \( \psi \) such that for every database \( \text{db} \):

\[
\text{db} \in \text{CQA}(q) \iff \text{db satisfies } \psi
\]

\( \psi \), if it exists, is called a consistent first-order rewriting for \( q \).

Why is this interesting?

- If \( \text{CQA}(q) \) is first-order definable, then \( \text{CQA}(q) \) is in \( \mathbf{P} \) (even in \( \mathbf{AC}^0 \)).
- \( \psi \) can be encoded in SQL...
Example

<table>
<thead>
<tr>
<th>EMP</th>
<th>Name</th>
<th>Status</th>
<th>Dept</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ed</td>
<td>married</td>
<td>Toys</td>
</tr>
<tr>
<td></td>
<td>Ed</td>
<td>single</td>
<td>Toys</td>
</tr>
</tbody>
</table>

PRIMARY KEY(Name)

First repair:

- Ed married Toys

Second repair:

- Ed single Toys

\[ q = \exists x \exists z \text{EMP}(x, \text{married}, z) \]

\[ \psi = \exists x \exists z (\text{EMP}(x, \text{married}, z) \land \forall y \forall z' (\text{EMP}(x, y, z') \rightarrow y = \text{married})) \]
Example

<table>
<thead>
<tr>
<th>EMP</th>
<th>Name</th>
<th>Status</th>
<th>Dept</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ed</td>
<td>married</td>
<td>Toys</td>
</tr>
<tr>
<td></td>
<td>Ed</td>
<td>single</td>
<td>Toys</td>
</tr>
</tbody>
</table>

PRIMARY KEY(Name)

First repair:

<table>
<thead>
<tr>
<th>EMP</th>
<th>Name</th>
<th>Status</th>
<th>Dept</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ed</td>
<td>married</td>
<td>Toys</td>
</tr>
</tbody>
</table>

Second repair:

<table>
<thead>
<tr>
<th>EMP</th>
<th>Name</th>
<th>Status</th>
<th>Dept</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ed</td>
<td>single</td>
<td>Toys</td>
</tr>
</tbody>
</table>

\[ q = \exists x \exists z \text{EMP}(x, \text{married}, z) \]

\[ \psi = \exists x \exists z (\text{EMP}(x, \text{married}, z) \land \forall y \forall z' (\text{EMP}(x, y, z') \rightarrow y = \text{married}) \]

No matter how EMP\((x, \cdot, \cdot)\) is repaired.
Facts and Conjectures

- If $P \neq NP$, then

$$\text{CQA}(q) \text{ coNP-complete} \implies q \text{ has no consistent first-order rewriting.}$$
Facts and Conjectures

- If $P \neq NP$, then
  
  $\text{CQA}(q)$ coNP-complete $\implies q$ has no consistent first-order rewriting.

- The implication $\iff$ does not generally hold:

  Theorem 1. [Wij07] For $q = \exists x \exists y (R(x, y) \land R(y, c))$,

  - $q$ has no consistent first-order rewriting, and
  - $\text{CQA}(q)$ is in $P$. 

Facts and Conjectures

- If $P \neq NP$, then
  $$\text{CQA}(q) \text{ coNP-complete} \implies q \text{ has no consistent first-order rewriting.}$$

- The implication $\iff$ does not generally hold:
  **Theorem 1.** [Wij07] For $q = \exists x \exists y (R(x, y) \land R(y, c))$, 
  - $q$ has no consistent first-order rewriting, and
  - CQA($q$) is in $P$.

- **Conjecture 1.** For Boolean conjunctive queries $q$ in which every relation name occurs at most once,
  $$q \text{ has no consistent first-order rewriting} \implies \text{CQA}(q) \text{ is coNP-complete.}$$
Table of Contents

- Consistent First-order Rewriting
- The Class $C_{rooted}$
- Deciding $C_{rooted}$
From now on, \( \exists \)-quantification is implicitly understood.

An atom \( R_i(x_i, y_i) \) is reifiable in

\[
q = R_1(x_1, y_1) \land \cdots \land R_m(x_m, y_m)
\]

if for every database \( db \),

if every repair of \( db \) satisfies \( q \), then every repair of \( db \) also satisfies \( \theta(q) \) for some valuation \( \theta \) over \( x_i \).
Reifiable Atom

- From now on, $\exists$-quantification is implicitly understood.
- An atom $R_i(\bar{x}_i, \bar{y}_i)$ is reifiable in
  \[ q = R_1(\bar{x}_1, \bar{y}_1) \land \cdots \land R_m(\bar{x}_m, \bar{y}_m) \]
  if for every database $db$,
  if every repair of $db$ satisfies $q$, then every repair of $db$ also satisfies $\theta(q)$ for some valuation $\theta$ over $\bar{x}_i$.
- Notice that $\theta$ depends on $db$.
- (A valuation over $\bar{x}_i$ is a mapping that maps every variable in $\bar{x}_i$ to a constant, and that is the identity on variables not in $\bar{x}_i$ and on constants.)
Reifiable Atom: Example

- $R(y)$ is not reifiable in $q = R(y) \land S(x, y)$.

Let $db = \{ R(a), R(b), S(c, a), S(c, b) \}$ with two repairs:

$$\text{rep}_1 = \{ R(a), R(b), S(c, a) \}$$
$$\text{rep}_2 = \{ R(a), R(b), S(c, b) \}$$

- Both repairs satisfy $q$.

- There is no valuation $\theta$ over $y$ such that both repairs satisfy $\theta(q)$. In particular,
  - $\text{rep}_2$ falsifies $\theta_a(q)$ with $\theta_a = \{ y \mapsto a \}$
  - $\text{rep}_1$ falsifies $\theta_b(q)$ with $\theta_b = \{ y \mapsto b \}$

- On the other hand, it is not hard to see that $S(x, y)$ is reifiable in $q = R(y) \land S(x, y)$. 
Ordered Boolean Conjunctive Queries

- A Boolean conjunctive query is just a finite set of atoms (the $\exists$-quantification is understood).

- An ordered Boolean conjunctive query is a finite sequence of atoms: $\langle R_1(x_1, y_1), \ldots, R_m(x_m, y_m) \rangle$.

- All definitions given for (non-ordered) queries apply to ordered queries (simply omit the order).

- The term rule is a shorthand for Boolean conjunctive query.
Rooted ordered rules are recursively defined as follows:

1. The empty rule is rooted.
2. A nonempty ordered rule

\[ q = \langle R_1(x_1, y_1), R_2(x_2, y_2), \ldots, R_n(x_n, y_n) \rangle \]

\((n \geq 1)\) is rooted if \(R_1(x_1, y_1)\) is reifiable in \(q\) and for every valuation \(\theta\) over \(x_1 y_1\), the shorter rule

\[ \langle \theta(R_2(x_2, y_2)), \ldots, \theta(R_n(x_n, y_n)) \rangle \]

is rooted.
Significant Result

- We say that a (non-ordered) rule is rooted if it is rooted for some linear ordering of its atoms.
- The class of rooted rules is denoted by $C_{\text{rooted}}$. 
We say that a (non-ordered) rule is rooted if it is rooted for some linear ordering of its atoms.

The class of rooted rules is denoted by $C_{\text{rooted}}$.

**Theorem 2.** [Wij07] If $q \in C_{\text{rooted}}$, then $\text{CQA}(q)$ is first-order definable.

In other words, every rooted rule has a consistent first-order rewriting.
We say that a (non-ordered) rule is rooted if it is rooted for some linear ordering of its atoms.

The class of rooted rules is denoted by $C_{\text{rooted}}$.

Theorem 2. [Wij07] If $q \in C_{\text{rooted}}$, then $\text{CQA}(q)$ is first-order definable. In other words, every rooted rule has a consistent first-order rewriting.

$C_{\text{forest}} \subsetneq C_{\text{rooted}}$ (see [FM07] for $C_{\text{forest}}$).
We say that a (non-ordered) rule is rooted if it is rooted for some linear ordering of its atoms.

The class of rooted rules is denoted by $C_{\text{rooted}}$.

**Theorem 2.** [Wij07] If $q \in C_{\text{rooted}}$, then $\text{CQA}(q)$ is first-order definable.

In other words, every rooted rule has a consistent first-order rewriting.

$C_{\text{forest}} \subsetneq C_{\text{rooted}}$ (see [FM07] for $C_{\text{forest}}$).

**Conjecture 2.** If a rule is not rooted, then it has no consistent first-order rewriting.

**Theorem 3.** [Wij09] If a rule $R(\vec{x}, \vec{y}), S(\vec{u}, \vec{w})$ with $R \neq S$ is not rooted, then it has no consistent first-order rewriting.
For example,

\[ q = \langle R(x, y), S(y, a) \rangle \] is rooted (to be shown).

\[ R(x, y_1), S(y_2, z), y_1 = y_2, z = a \]

Consistent first-order rewriting:

\[
\psi = \exists x \exists y'_1 (R(x, y'_1) \land \forall y_1 (R(x, y_1) \rightarrow \\
\exists y_2 \exists z' (S(y_2, z') \land \forall z (S(y_2, z) \rightarrow y_1 = y_2 \land z = a)))
\]

Special care if the same relation name occurs more than once.
Rewrite Function: Example

- For example,
  - $q = \langle R(x, y), S(y, a) \rangle$ is rooted (to be shown).
  - $R(x, y_1), S(y_2, z), y_1 = y_2, z = a$

- Consistent first-order rewriting:

  **Reifiability**

  $$
  \psi = \exists x \exists y'_1 (R(x, y'_1)) \land \forall y_1 (R(x, y_1) \rightarrow \\
  \exists y_2 \exists z' (S(y_2, z')) \land \forall z (S(y_2, z) \rightarrow y_1 = y_2 \land z = a))
  $$

- Special care if the same relation name occurs more than once.
For example,

\[ q = \langle R(x, y), S(y, a) \rangle \] is rooted (to be shown).

\[ R(x, y_1), S(y_2, z), y_1 = y_2, z = a \]

Consistent first-order rewriting:

All ways of repairing

\[ \psi = \exists x \exists y_1' (R(x, y_1') \land \forall y_1 (R(x, y_1) \rightarrow \exists y_2 \exists z' (S(y_2, z') \land \forall z (S(y_2, z) \rightarrow y_1 = y_2 \land z = a))) \]

Special care if the same relation name occurs more than once.

Symposium on Consistency Handling in Relational Databases, Leuven, 19 December 2008 – p.16/32
Table of Contents

- Consistent First-order Rewriting
- The Class $C_{rooted}$
- Deciding $C_{rooted}$
Is $C_{\text{rooted}}$ Decidable?

Notice that $C_{\text{rooted}}$ is a semantic class.
Is $C_{\text{rooted}}$ Decidable?

- Notice that $C_{\text{rooted}}$ is a **semantic** class.
- The crux [to decidability of $C_{\text{rooted}}$] is the **Reifiability Problem**:
  
  Given a rule $q$ and an atom $R(\overline{x}, \overline{y}) \in q$, is $R(\overline{x}, \overline{y})$ reifiable in $q$?
Is $C_{\text{rooted}}$ Decidable?

- Notice that $C_{\text{rooted}}$ is a semantic class.
- The crux [to decidability of $C_{\text{rooted}}$] is the Reifiability Problem:
  
  Given a rule $q$ and an atom $R(\overline{x}, \overline{y}) \in q$, is $R(\overline{x}, \overline{y})$ reifiable in $q$?

- Moreover, Theorem 4.[Wij09] Let $q$ be an ordered rule in which no relation name occurs more than once. Let $X$ be a set of variables.
  If $\theta(q)$ is rooted for some valuation $\theta$ over $X$, then $\mu(q)$ is rooted for every valuation $\mu$ over $X$. 

Outline

- We will define the syntactic construct of reifiability-attack, which relies on
  - Key-closure
  - Join tree [BFMY83]
We will define the syntactic construct of reifiability-attack, which relies on
  
  Key-closure
  
  Join tree [BFMY83]

What’s in a name?
  
  A reifiability-attack against an atom $A$ implies that $A$ is not reifiable (under some mild additional condition).
  
  An atom without reifiability-attack against it, is reifiable (always).
Key-closure: Motivating Example

- Let $q = R(x, y) \land S(x, y)$.
- Let $db$ be a database such that every repair of $db$ satisfies $q$. 
Let $q = R(x, y) \land S(x, y)$.

Let $db$ be a database such that every repair of $db$ satisfies $q$.

Then, we can assume constants $a, b$ such that:
- $R(a, b), S(a, b) \in db$; and
- whenever $c \neq b$, then $R(a, c) \not\in db$ and $S(a, c) \not\in db$. 
Let $q = R(x, y) \land S(x, y)$.

Let $db$ be a database such that every repair of $db$ satisfies $q$.

Then, we can assume constants $a, b$ such that:
- $R(a, b), S(a, b) \in db$; and
- whenever $c \neq b$, then $R(a, c) \notin db$ and $S(a, c) \notin db$.

Then, there is a constant $a$ such that every repair of $db$ satisfies $\theta_a(q)$, where $\theta_a = \{x \mapsto a\}$. 
Key-closure: Motivating Example

Let \( q = R(x, y) \land S(x, y) \).

Let \( db \) be a database such that every repair of \( db \) satisfies \( q \).

Then, we can assume constants \( a, b \) such that:
- \( R(a, b), S(a, b) \in db \); and
- whenever \( c \neq b \), then \( R(a, c) \notin db \) and \( S(a, c) \notin db \).

Then, there is a constant \( a \) such that every repair of \( db \) satisfies \( \theta_a(q) \), where \( \theta_a = \{x \mapsto a\} \).

Moreover, for all valuations \( \theta_1, \theta_2 \) over \( \{x, y\} \),
- if \( \theta_1(q), \theta_2(q) \subseteq db \) and \( \theta_1(x) = \theta_2(x) = a \),
- then \( \theta_1(y) = \theta_2(y) \).

We will write \( y \in [{\{x\}}]^+ \).
Key-closure: Definition

- Relative to a rule \( q \) without self join (i.e. without repeated relation names).
- \( \text{vars}(\vec{x}) \) is the set of variables that occur in \( \vec{x} \).
Key-closure: Definition

Relative to a rule \( q \) without self join (i.e. without repeated relation names).

\( \text{vars}(\vec{x}) \) is the set of variables that occur in \( \vec{x} \).

Let \( X \) be a set of variables. The key-closure of \( X \), denoted \( [X]^+ \), is the smallest set satisfying:

\( X \subseteq [X]^+ \); and

for every \( R(\vec{x}, \vec{y}), S(\vec{u}, \vec{w}) \in q \) such that \( R \neq S \),

if \( \text{vars}(\vec{x}) = \text{vars}(\vec{u}) \subseteq [X]^+ \),

then \( \text{vars}(\vec{y}) \cap \text{vars}(\vec{w}) \subseteq [X]^+ \).
Key-closure: Definition

Relative to a rule $q$ without self join (i.e. without repeated relation names).

vars($\bar{x}$) is the set of variables that occur in $\bar{x}$.

Let $X$ be a set of variables. The key-closure of $X$, denoted $[X]^+$, is the smallest set satisfying:

$X \subseteq [X]^+$; and

for every $R(\bar{x}, \bar{y}), S(\bar{u}, \bar{w}) \in q$ such that $R \neq S$,

- if $\text{vars}(\bar{x}) = \text{vars}(\bar{u}) \subseteq [X]^+$,
- then $\text{vars}(\bar{y}) \cap \text{vars}(\bar{w}) \subseteq [X]^+$.

For example, for

$R_0(z), R_1(x, y, z), R_2(x, y, u), R_3(x, y), R_4(x, y, u)$

we have $\{x\}^+ = \{x, y\}^+ = \{x, y, u\}$. 
A rule $q$ is \textit{acyclic} if it has a join tree [BFMY83].

A \textit{join tree} for a rule $q$ is an undirected tree whose vertices are the atoms of $q$ such that:

\textbf{Connectedness Condition:} whenever the same variable $z$ occurs in two atoms $R_i(\overline{x}_i, \overline{y}_i)$ and $R_j(\overline{x}_j, \overline{y}_j)$, then $z$ occurs in each atom on the unique path linking $R_i(\overline{x}_i, \overline{y}_i)$ and $R_j(\overline{x}_j, \overline{y}_j)$.

It is common to label each edge with the set of variables that occur in both end points.
Join Tree: Example

\begin{center}
\begin{tikzpicture}
  \node (r0) at (0,0) {$R_0(x, y)$};
  \node (r1) at (0,-2) {$R_1(x, y)$};
  \node (r2) at (-1,-4) {$R_2(y, z)$};
  \node (r3) at (1,-4) {$R_3(y, a)$};
  \node (r4) at (0,-6) {$R_4(z, w)$};
  \node (x) at (0,-3) {$\{x, y\}$};
  \node (y1) at (0,-5) {$\{y\}$};
  \node (y2) at (1,-5) {$\{y\}$};
  \node (z) at (0,-7) {$\{z\}$};

  \draw (r0) -- (x);
  \draw (r1) -- (y1);
  \draw (r1) -- (y2);
  \draw (r2) -- (z);
  \draw (r3) -- (z);
  \draw (r4) -- (z);
\end{tikzpicture}
\end{center}
Let $q$ be an acyclic query. Let $\tau$ be a join tree for $q$.

A reifiability-attack against an atom $R(\vec{x}, \vec{y}) \in q$ is a path $\pi$ in $\tau$ from $R(\vec{x}, \vec{y})$ to some atom $S(\vec{u}, \vec{w})$ such that:

- $\text{vars}(\vec{x}) \notin \text{vars}^+(\vec{u})$; and
- for each label $L$ on $\pi$, we have $L \notin \text{vars}^+(\vec{u})$. 

---

**Reifiability-attack**
Let $q$ be an acyclic query. Let $\tau$ be a join tree for $q$.

A reifiability-attack against an atom $R(\vec{x}, \vec{y}) \in q$ is a path $\pi$ in $\tau$ from $R(\vec{x}, \vec{y})$ to some atom $S(\vec{u}, \vec{w})$ such that:

- $\text{vars}(\vec{x}) \not\subseteq \text{vars}^{+}(\vec{u})$; and
- for each label $L$ on $\pi$, we have $L \not\subseteq \text{vars}^{+}(\vec{u})$.

Thus, a reifiability-attack against $R(\vec{x}, \vec{y})$ is a path that starts from $R(\vec{x}, \vec{y})$ such that:

$\text{vars}(\vec{x})$ and each label on the path are not contained in the key-closure of the primary key of the last atom on the path.
Reifiability-attack: Example

The red path is a reifiability-attack against $R_4(z, w)$, because:

- $\text{vars}(z) \nsubseteq \text{vars}^+(x) = \{x\}$, and
- $\{z\}, \{y\} \nsubseteq \text{vars}^+(x)$. 

Symposium on Consistency Handling in Relational Databases, Leuven, 19 December 2008 – p.25/32
The blue path is not a reifiability-attack against $R_4(z, w)$, because $\{y\} \subseteq \text{vars}^+(\underline{x}) = \{x, y\}$. 
Absence of Reifiability-attack

Theorem 5. [Wij09] Let $q$ be an acyclic rule without self join. Let $\tau$ be a join tree for $q$. Let $A \in q$. If $\tau$ contains no reifiability-attack against $A$, then $A$ is reifiable in $q$.

We also showed that, under some (mild) additional condition, a reifiability-attack against $A$ implies that $A$ is not reifiable.
Corollary 1. A rule \( q \) without self join is rooted if it has a directed rooted join tree \( \tau \) such that whenever the atom \( R_i(\overline{x}_i, \overline{y}_i) \) is the parent of \( R_j(\overline{x}_j, \overline{y}_j) \), then either

- \( \text{vars}(\overline{x}_i) \subseteq \text{vars}(\overline{x}_j) \), or
- \( L \subseteq \text{vars}(\overline{x}_j) \),

where \( L \) is the label on the edge between \( R_i(\overline{x}_i, \overline{y}_i) \) and \( R_j(\overline{x}_j, \overline{y}_j) \).
Corollary 1. A rule $q$ without self join is rooted if it has a directed rooted join tree $\tau$ such that whenever the atom $R_i(\overline{x}_i, \overline{y}_i)$ is the parent of $R_j(\overline{x}_j, \overline{y}_j)$, then either

- $\text{vars}(\overline{x}_i) \subseteq \text{vars}(\overline{x}_j)$, or
- $L \subseteq \text{vars}(\overline{x}_j)$,

where $L$ is the label on the edge between $R_i(\overline{x}_i, \overline{y}_i)$ and $R_j(\overline{x}_j, \overline{y}_j)$.

This works essentially because no “downward” path can be a reifiability-attack against the first atom on the path.
Corollary 1. A rule \( q \) without self join is rooted if it has a directed rooted join tree \( \tau \) such that whenever the atom \( R_i(x_i, y_i) \) is the parent of \( R_j(x_j, y_j) \), then either

- \( \text{vars}(x_i) \subseteq \text{vars}(x_j) \), or
- \( L \subseteq \text{vars}(x_j) \),

where \( L \) is the label on the edge between \( R_i(x_i, y_i) \) and \( R_j(x_j, y_j) \).

This works essentially because no “downward” path can be a reifiability-attack against the first atom on the path.

The tree components of a Boolean \( C_{forest} \) query [FM07] are special cases of this. Covers many natural queries.
Application: Example

The blue path is not a reifiability-attack against $R(x, y)$, because $\text{vars}(x) \subseteq \text{vars}^+(x, u) = \{x, u\}$.

The green path is not a reifiability-attack against $R(x, y)$, because $\{y\} \subseteq \text{vars}^+(y, z) = \{y, z\}$. 
The idea of certain (or consistent) query answers on inconsistent databases was proposed in [ABC99]. That paper also brought up the idea of consistent query rewriting.

Consistent first-order query rewriting of conjunctive queries under primary keys was studied by Fuxman and Miller [FM05, FM07]. Extensions can be found in [GLRR05, LRR06].
References


