Principles of Managing Uncertain Data

Lecture 8: Consistent Query Answering
Table of Contents

1. Introduction
2. Trichotomy Theorem
3. Attacks
4. Refined Trichotomy
5. FO Rewriting with SQL
Many thanks to Jef Wijsen for helping with the slides!
# Table of Contents

1. Introduction
2. Trichotomy Theorem
3. Attacks
4. Refined Trichotomy
5. FO Rewriting with SQL
Previous Lecture

- Defined *inconsistent databases* and *repairs*
- Defined *Consistent Query Answering (CQA)*
Previous Lecture

- Defined *inconsistent databases* and *repairs*
- Defined *Consistent Query Answering* (CQA)
- Saw a schema with primary-key constraints and a CQ where:
  - CQA can be translated into a formula in *First Order Logic* (FO) over the inconsistent instance
    - Hence, computable in polynomial time
  - CQA is *coNP-hard*
  - CQA *cannot be phrased in FO* over the inconsistent instance, but is still *computable in polynomial time*
This Lecture

- We will focus on schemas with primary-key constraints, and CQs *without self joins*
  - That is, CQs where each relation occurs at most once
This Lecture

- We will focus on schemas with primary-key constraints, and CQs without self joins
  - That is, CQs where each relation occurs at most once
- We will learn a recent result that shows how to distinguish between the three cases
This Lecture

- We will focus on schemas with primary-key constraints, and CQs \textit{without self joins}
  - That is, CQs where each relation occurs at most once
- We will learn a recent result that shows how to distinguish between the three cases
- Such a result is called a \textit{trichotomy}, since it classifies all cases into three pairwise-disjoint categories
We will focus on schemas with primary-key constraints, and CQs *without self joins*

- That is, CQs where each relation occurs at most once

We will learn a recent result that shows how to distinguish between the three cases

Such a result is called a *trichotomy*, since it classifies all cases into three pairwise-disjoint categories

We will see how to rewrite CQA into SQL in the case of FO rewritability
Table of Contents

1 Introduction
2 Trichotomy Theorem
3 Attacks
4 Refined Trichotomy
5 FO Rewriting with SQL
In this lecture we consider only schemas $S = (\mathcal{R}, \Sigma)$ such that $\Sigma$ consists of primary keys.
In this lecture we consider only schemas $\mathcal{S} = (\mathcal{R}, \Sigma)$ such that $\Sigma$ consists of primary keys.

That is:

- For every relation name $R \in \mathcal{R}$ there is a unique key constraint $R : X \rightarrow Y$ in $\Sigma$.
- There are no other constraints in $\Sigma$. 

Note: “no key” is the same as “left-hand side contains all attributes.” In our examples we underline the key attributes.
In this lecture we consider only schemas $S = (\mathcal{R}, \Sigma)$ such that $\Sigma$ consists of primary keys.

That is:

- For every relation name $R \in \mathcal{R}$ there is a unique key constraint $R : X \to Y$ in $\Sigma$.
- There are no other constraints in $\Sigma$.

Note: “no key” is the same as “left-hand side contains all attributes”.
In this lecture we consider only schemas $S = (\mathcal{R}, \Sigma)$ such that $\Sigma$ consists of primary keys.

That is:

- For every relation name $R \in \mathcal{R}$ there is a unique key constraint $R : X \rightarrow Y$ in $\Sigma$.
- There are no other constraints in $\Sigma$.

Note: “no key” is the same as “left-hand side contains all attributes”.

In our examples we underline the key attributes.

For instance, if $\mathcal{R}$ contains $R(A, B, C, D)$ then $R(x, y, z, w)$ means that $\Sigma$ contains the key constraint $R : AB \rightarrow CD$. 
Recall: a CQ over a signature $\mathcal{R}$ is a query of the form:

$$Q(x) :- \exists y[\varphi_1(x,y) \land \cdots \land \varphi_k(x,y)]$$

where each $\varphi_k(x,y)$ is an atomic query

Each $\varphi_i$ is called an *atom* of $Q$
CQs without Self Joins

- Recall: a CQ over a signature $\mathcal{R}$ is a query of the form:

$$Q(\mathbf{x}) \leftarrow \exists \mathbf{y} [\varphi_1(\mathbf{x}, \mathbf{y}) \land \cdots \land \varphi_k(\mathbf{x}, \mathbf{y})]$$

where each $\varphi_k(\mathbf{x}, \mathbf{y})$ is an atomic query

- Each $\varphi_i$ is called an atom of $Q$

- We say that $Q$ has no self joins if no two distinct atoms use the same relation name
CQs without Self Joins

- Recall: a CQ over a signature $\mathcal{R}$ is a query of the form:

$$Q(x) \leftarrow \exists y [\varphi_1(x, y) \land \cdots \land \varphi_k(x, y)]$$

where each $\varphi_k(x, y)$ is an atomic query

- Each $\varphi_i$ is called an atom of $Q$

- We say that $Q$ has no self joins if no two distinct atoms use the same relation name

- We say that $Q$ is Boolean if $x$ is empty; in that case, $Q$ is either true or false on a given instance $I$
Consistent Answers

**Definition (Consistent Answers)**

Let \( S = (R, \Sigma) \) be a schema, \( Q \) a query over \( S \), and \( I \) an inconsistent instance over \( S \). A tuple \( a \) is a **consistent answer** if \( a \in Q(J) \) for every repair \( J \). We denote by \( \text{Consistent}_Q^{\Sigma}(I) \) the set of all consistent answers. Hence, we have:

\[
\text{Consistent}_Q^{\Sigma}(I) = \bigcap_{J \in \text{Repairs}_\Sigma(I)} Q(J)
\]
Recalling Data Complexity

- Recall: in *data complexity* we fix the schema and query, and only the instance $I$ is considered input.
Recalling Data Complexity

- Recall: in data complexity we fix the schema and query, and only the instance $I$ is considered input.
- Effectively, every schema $S$ and query $Q$ define a separate computational problem $P_{S,Q}$. 

**Theorem** [KW15]

Let $S = (\mathcal{R}, \Sigma)$ be a schema, such that $\Sigma$ consists of primary keys. Let $Q$ be a CQ without self joins. Assume that $P \neq NP$. Then exactly one of the following is true.

1. Consistent $Q$ can be formulated as a query in FO (hence, computable in polynomial time).
2. Consistent $Q$ cannot be formulated as a query in FO, but is still computable in polynomial time.
3. Testing whether Consistent $Q$ is empty is NP-complete.

Moreover, we can compute in polynomial time (in $S$ and $Q$) in which case we are.
**Theorem [KW15]**

Let $S = (\mathcal{R}, \Sigma)$ be a schema, such that $\Sigma$ consists of primary keys. Let $Q$ be a CQ without self joins. Assume that $P \neq NP$. Then exactly one of the following is true.

1. $\text{Consistent}_S^Q$ can be formulated as a query in FO (hence, computable in polynomial time).
**Theorem [KW15]**

Let $S = (R, \Sigma)$ be a schema, such that $\Sigma$ consists of primary keys. Let $Q$ be a CQ without self joins. Assume that $P \neq NP$. Then exactly one of the following is true.

1. $\text{Consistent}_{\Sigma}^{Q}$ can be formulated as a query in FO (hence, computable in polynomial time).

2. $\text{Consistent}_{\Sigma}^{Q}$ cannot be formulated as a query in FO, but is still computable in polynomial time.
**Trichotomy Theorem**

**Theorem [KW15]**

Let $\mathcal{S} = (\mathcal{R}, \Sigma)$ be a schema, such that $\Sigma$ consists of primary keys. Let $Q$ be a CQ without self joins. Assume that $P \neq NP$. Then exactly one of the following is true.

1. $\text{Consistent}^Q_{\Sigma}$ can be formulated as a query in FO (hence, computable in polynomial time).

2. $\text{Consistent}^Q_{\Sigma}$ cannot be formulated as a query in FO, but is still computable in polynomial time.

3. Testing whether $\text{Consistent}^Q_{\Sigma}$ is empty is NP-complete.
Trichotomy Theorem

**Theorem [KW15]**

Let $S = (\mathcal{R}, \Sigma)$ be a schema, such that $\Sigma$ consists of primary keys. Let $Q$ be a CQ without self joins. Assume that $P \neq NP$. Then exactly one of the following is true.

1. $\text{Consistent}_{Q}^{\Sigma}$ can be formulated as a query in FO (hence, computable in polynomial time).
2. $\text{Consistent}_{Q}^{\Sigma}$ cannot be formulated as a query in FO, but is still computable in polynomial time.
3. Testing whether $\text{Consistent}_{Q}^{\Sigma}$ is empty is NP-complete.

Moreover, we can compute in polynomial time (in $S$ and $Q$) in which case we are.
Historical Notes I

- **2005:** Fuxman and Miller [FM05] claim a dichotomy for a class of conjunctive queries without self joins
  - A flaw in their proof and result discovered by Wijsen [Wij10b]
- **2010:** Wijsen [Wij10a] establishes a dichotomy in FO rewritability for *acyclic CQs without self joins*
- **2012:** Kolaitis and Pema [KP12] prove a dichotomy (P vs coNP-complete) for *CQs with two atoms and no self joins*
2013: Fontaine [Fon13] establishes an explanation on why it is difficult to establish dichotomies for (U)CQs with self joins. Basically, it entails solving a long standing open problem.

2014: Koutris and Suciu [KS14] prove a dichotomy for CQs without self joins, where every relation is binary (with a key).

2015: Koutris and Wijsen [KW15] prove a trichotomy for all CQs without self joins.

That is, the trichotomy we learn here.
<table>
<thead>
<tr>
<th>#</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
</tr>
<tr>
<td>2</td>
<td>Trichotomy Theorem</td>
</tr>
<tr>
<td>3</td>
<td>Attacks</td>
</tr>
<tr>
<td>4</td>
<td>Refined Trichotomy</td>
</tr>
<tr>
<td>5</td>
<td>FO Rewriting with SQL</td>
</tr>
</tbody>
</table>
Throughout this section, we fix a schema $S = (\mathcal{R}, \Sigma)$ and a CQ $Q$. 
Setup

- Throughout this section, we fix a schema \( \mathcal{S} = (\mathcal{R}, \Sigma) \) and a CQ \( Q \)
  - \( \Sigma \) consists of primary keys (one for each relation)
  - \( Q \) has no self joins
Throughout this section, we fix a schema $\mathcal{S} = (\mathcal{R}, \Sigma)$ and a CQ $Q$:

- $\Sigma$ consists of primary keys (one for each relation)
- $Q$ has no self joins

Recall that for such $\Sigma$, a *repair* is a maximal consistent subset of $I$. 
Throughout this section, we fix a schema $\mathcal{S} = (\mathcal{R}, \Sigma)$ and a CQ $Q$:
- $\Sigma$ consists of primary keys (one for each relation)
- $Q$ has no self joins

Recall that for such $\Sigma$, a repair is a maximal consistent subset of $I$.

We first assume that $Q$ is Boolean (that is, there are no variables in the head):
- Hence, the goal is to determine whether $Q$ is true in every repair.
We denote by:

- $\text{Atoms}(Q)$ the set of atoms of $Q$
- $\text{Var}(Q)$ the set of all the variables of $Q$
- $\alpha_R$ the atom of $Q$ over the relation name $R$
- $R_\alpha$ the relation name of the atom $\alpha$
Notation

- We denote by:
  - \( \text{Atoms}(Q) \) the set of atoms of \( Q \)
  - \( \text{Var}(Q) \) the set of all the variables of \( Q \)
  - \( \alpha_R \) the atom of \( Q \) over the relation name \( R \)
  - \( R_\alpha \) the relation name of the atom \( \alpha \)

- For \( \alpha \in \text{Atoms}(Q) \), we denote by:
  - \( \text{Var}(\alpha) \) the variables that occur in \( \alpha \)
  - \( \text{KVar}(\alpha) \) the variables that occur in key attributes of \( R_\alpha \)
Example

\[ Q() \defeq R(x, a, y), S(x, z, w), T(z, y, u), U(b, z) \]
Example

\[ Q() \leftarrow R(x, a, y), S(x, z, w), T(z, y, u), U(b, z) \]

- \( \text{Atoms}(Q) = \{ R(x, a, y), S(x, z, w), T(z, y, u), U(b, z) \} \)
Example

\[ Q() :- R(x, a, y), S(x, z, w), T(z, y, u), U(b, z) \]

- Atoms(\( Q \)) = \{R(x, a, y), S(x, z, w), T(z, y, u), U(b, z)\}
- Var(\( Q \)) = \{x, y, z, w, u\}
Example

\[ Q() \leftarrow R(x, a, y), S(x, z, w), T(z, y, u), U(b, z) \]

- \( \text{Atoms}(Q) = \{ R(x, a, y), S(x, z, w), T(z, y, u), U(b, z) \} \)
- \( \text{Var}(Q) = \{ x, y, z, w, u \} \)
- \( \alpha_S = S(x, z, w) \)
Example

\[ Q() \leftarrow R(x, a, y), S(x, z, w), T(z, y, u), U(b, z) \]

- \( \text{Atoms}(Q) = \{ R(x, a, y), S(x, z, w), T(z, y, u), U(b, z) \} \)
- \( \text{Var}(Q) = \{ x, y, z, w, u \} \)
- \( \alpha_S = S(x, z, w) \)
- \( \alpha = \alpha_R = R(x, a, y) \Rightarrow R_{\alpha} = R, \ Var(\alpha) = \{ x, y \}, \ KVar(\alpha) = \{ x \} \)
Example

\[ Q() : - \ R(x, a, y), S(x, z, w), T(z, y, u), U(b, z) \]

- \(\text{Atoms}(Q) = \{R(x, a, y), S(x, z, w), T(z, y, u), U(b, z)\}\)
- \(\text{Var}(Q) = \{x, y, z, w, u\}\)
- \(\alpha_S = S(x, z, w)\)
- \(\alpha = \alpha_R = R(x, a, y) \implies R_\alpha = R, \ \text{Var}(\alpha) = \{x, y\}, \ \text{KVar}(\alpha) = \{x\}\)
- \(\alpha = S(x, z, w) \implies R_\alpha = S, \ \text{Var}(\alpha) = \{x, z, w\}, \ \text{KVar}(\alpha) = \{x\}\)
Example

\[ Q() : - R(x, a, y), S(x, z, w), T(z, y, u), U(b, z) \]

- \( \text{Atoms}(Q) = \{ R(x, a, y), S(x, z, w), T(z, y, u), U(b, z) \} \)
- \( \text{Var}(Q) = \{ x, y, z, w, u \} \)
- \( \alpha_s = S(x, z, w) \)
- \( \alpha = \alpha_R = R(x, a, y) \Rightarrow R_\alpha = R, \text{Var}(\alpha) = \{ x, y \}, \text{KVar}(\alpha) = \{ x \} \)
- \( \alpha = S(x, z, w) \Rightarrow R_\alpha = S, \text{Var}(\alpha) = \{ x, z, w \}, \text{KVar}(\alpha) = \{ x \} \)
- We denote constants by non-italic letters from the beginning of the alphabet (e.g., \( a \) and \( b \)), as opposed to variables (e.g., \( x \) and \( y \)).
We define the following set of functional dependencies (FDs):

\[
\text{FD}(Q) \overset{\text{def}}{=} \{ \text{KVar}(\alpha) \rightarrow \text{Var}(\alpha) \mid \alpha \in \text{Atoms}(Q) \}
\]
FDs among Variables

- We define the following set of functional dependencies (FDs):

\[ \text{FD}(Q) \overset{\text{def}}{=} \{ \text{KVar}(\alpha) \rightarrow \text{Var}(\alpha) \mid \alpha \in \text{Atoms}(Q) \} \]

- \( \text{FD}^+(Q) \) denotes the set of all FDs over \( \text{Var}(Q) \) that are logically implied from \( \text{FD}(Q) \)
FDs among Variables

- We define the following set of functional dependencies (FDs):
  \[ FD(Q) \overset{\text{def}}{=} \{ KVar(\alpha) \rightarrow \text{Var}(\alpha) \mid \alpha \in \text{Atoms}(Q) \} \]

- \( FD^+(Q) \) denotes the set of all FDs over \( \text{Var}(Q) \) that are logically implied from \( FD(Q) \)

- Equivalently (by Armstrong’s axioms), \( FD^+(Q) \) is obtained from \( FD(Q) \) by repeatedly applying the following rules:
FDs among Variables

- We define the following set of functional dependencies (FDs):

\[ \text{FD}(Q) \overset{\text{def}}{=} \{ \text{KVar}(\alpha) \rightarrow \text{Var}(\alpha) \mid \alpha \in \text{Atoms}(Q) \} \]

- \( \text{FD}^+(Q) \) denotes the set of all FDs over \( \text{Var}(Q) \) that are logically implied from \( \text{FD}(Q) \)

- Equivalently (by Armstrong’s axioms), \( \text{FD}^+(Q) \) is obtained from \( \text{FD}(Q) \) by repeatedly applying the following rules:
  - \( X \rightarrow X' \) whenever \( X' \subseteq X \) (reflexivity)
We define the following set of functional dependencies (FDs):

\[
\text{FD}(Q) \overset{\text{def}}{=} \{ K\text{Var}(\alpha) \to \text{Var}(\alpha) \mid \alpha \in \text{Atoms}(Q) \}
\]

- \(\text{FD}^+(Q)\) denotes the set of all FDs over \(\text{Var}(Q)\) that are logically implied from \(\text{FD}(Q)\)
- Equivalently (by Armstrong’s axioms), \(\text{FD}^+(Q)\) is obtained from \(\text{FD}(Q)\) by repeatedly applying the following rules:
  - \(X \to X'\) whenever \(X' \subseteq X\) (reflexivity)
  - If \(X \to Y\) and \(Y \to Z\), then \(X \to Z\) (transitivity)
We define the following set of functional dependencies (FDs):

\[ \text{FD}(Q) \overset{\text{def}}{=} \{ \text{KVar}(\alpha) \rightarrow \text{Var}(\alpha) \mid \alpha \in \text{Atoms}(Q) \} \]

- \( \text{FD}^+(Q) \) denotes the set of all FDs over \( \text{Var}(Q) \) that are logically implied from \( \text{FD}(Q) \)

- Equivalently (by Armstrong’s axioms), \( \text{FD}^+(Q) \) is obtained from \( \text{FD}(Q) \) by repeatedly applying the following rules:
  - \( X \rightarrow X' \) whenever \( X' \subseteq X \) (reflexivity)
  - If \( X \rightarrow Y \) and \( Y \rightarrow Z \), then \( X \rightarrow Z \) (transitivity)
  - If \( X \rightarrow Y \), then \( X \cup Z \rightarrow Y \cup Z \) (augmentation)
Example

\[ Q() := R(x, a, y), S(x, z, w), T(z, y, u), U(b, z) \]

- \( \text{FD}(Q) = \{ x \rightarrow y, x \rightarrow zw, zy \rightarrow u, \emptyset \rightarrow z \} \)
Example

\[ Q() :\neg R(x, a, y), S(x, z, w), T(z, y, u), U(b, z) \]

- \( \text{FD}(Q) = \{ x \rightarrow y, x \rightarrow zw, \emptyset \rightarrow z \} \)
- \( \text{FD}^+(Q) = \{ x \rightarrow yzwu, y \rightarrow zu, u \rightarrow u, \ldots \} \cup \text{FD}(Q) \)
For $\alpha \in \text{Atoms}(Q)$ and $x \in \text{Var}(Q)$, we say that $x$ is an external dependent of $\alpha$ if $x$ is determined from the key of $\alpha$ even without $\alpha$; that is:

$$(K\text{Var}(\alpha) \rightarrow x) \in \text{FD}^+(Q \setminus \{\alpha\})$$
For $\alpha \in \text{Atoms}(Q)$ and $x \in \text{Var}(Q)$, we say that $x$ is an external dependent of $\alpha$ if $x$ is determined from the key of $\alpha$ even without $\alpha$; that is:

$$(\text{KVar}(\alpha) \rightarrow x) \in \text{FD}^+(Q \setminus \{\alpha\})$$

Observe that every $x \in \text{KVar}(\alpha)$ is an external dependent of $\alpha$. 
External Dependency

- For $\alpha \in \text{Atoms}(Q)$ and $x \in \text{Var}(Q)$, we say that $x$ is an 
  \textit{external dependent} of $\alpha$ if $x$ is determined from the key of $\alpha$
  even without $\alpha$; that is:

  \[
  (\text{KVar}(\alpha) \rightarrow x) \in \text{FD}^+(Q \setminus \{\alpha\})
  \]

- Observe that every $x \in \text{KVar}(\alpha)$ is an external dependent of $\alpha$

- Example: $Q() :- R(x, a, y), S(x, z, w), T(z, y, u), U(b, z)$
For $\alpha \in \text{Atoms}(Q)$ and $x \in \text{Var}(Q)$, we say that $x$ is an \textit{external dependent} of $\alpha$ if $x$ is determined from the key of $\alpha$ even without $\alpha$; that is:

$$(\text{KVar}(\alpha) \rightarrow x) \in \text{FD}^+(Q \setminus \{\alpha\})$$

Observe that every $x \in \text{KVar}(\alpha)$ is an external dependent of $\alpha$.

Example: $Q() :\leftarrow R(x, a, y), S(x, z, w), T(z, y, u), U(b, z)$

Which variables are external dependents of $\alpha_R$?
External Dependency

- For $\alpha \in \text{Atoms}(Q)$ and $x \in \text{Var}(Q)$, we say that $x$ is an **external dependent** of $\alpha$ if $x$ is determined from the key of $\alpha$ even without $\alpha$; that is:

$$\text{\(KVar(\alpha) \rightarrow x\)} \in \text{\(FD^+(Q \setminus \{\alpha\})\)}$$

- Observe that every $x \in KVar(\alpha)$ is an external dependent of $\alpha$.
- Example: $Q() :\text{\(- R(x, a, y), S(x, z, w), T(z, y, u), U(b, z)\)}$
  - *Which variables are external dependents of $\alpha_R$?* $x$, $z$, $w$
External Dependency

- For $\alpha \in \text{Atoms}(Q)$ and $x \in \text{Var}(Q)$, we say that $x$ is an external dependent of $\alpha$ if $x$ is determined from the key of $\alpha$ even without $\alpha$; that is:

$$\left(\text{KVar}(\alpha) \rightarrow x\right) \in \text{FD}^+(Q \setminus \{\alpha\})$$

- Observe that every $x \in \text{KVar}(\alpha)$ is an external dependent of $\alpha$
- Example: $Q() : \neg R(x, a, y), S(x, z, w), T(z, y, u), U(b, z)$
  - Which variables are external dependents of $\alpha_R$? $x, z, w$
  - Which variables are external dependents of $\alpha_S$?
For $\alpha \in \text{Atoms}(Q)$ and $x \in \text{Var}(Q)$, we say that $x$ is an *external dependent* of $\alpha$ if $x$ is determined from the key of $\alpha$ even without $\alpha$; that is:

$$(K\text{Var}(\alpha) \rightarrow x) \in \text{FD}^+(Q \setminus \{\alpha\})$$

Observe that every $x \in K\text{Var}(\alpha)$ is an external dependent of $\alpha$.

**Example:** $Q() :\neg R(x, a, y), S(x, z, w), T(z, y, u), U(b, z)$

- *Which variables are external dependents of $\alpha_R$?* $x, z, w$
- *Which variables are external dependents of $\alpha_S$?* $x, y, z, u$
Introduction
Trichotomy Theorem
Attacks
Refined Trichotomy
FO Rewriting with SQL
References

External Dependency

- For $\alpha \in \text{Atoms}(Q)$ and $x \in \text{Var}(Q)$, we say that $x$ is an external dependent of $\alpha$ if $x$ is determined from the key of $\alpha$ even without $\alpha$; that is:

$$\left( \text{KVar}(\alpha) \rightarrow x \right) \in \text{FD}^+(Q \setminus \{\alpha\})$$

- Observe that every $x \in \text{KVar}(\alpha)$ is an external dependent of $\alpha$
- Example: $Q() := R(x, a, y), S(x, z, w), T(z, y, u), U(b, z)$
  - Which variables are external dependents of $\alpha_R$? $x, z, w$
  - Which variables are external dependents of $\alpha_S$? $x, y, z, u$
- If $x$ is not an external dependent of $\alpha$, then we say that $x$ is externally independent of $\alpha$
Let $\alpha$ and $\gamma$ be two distinct atoms of $Q$

We say that $\alpha$ *attacks* $\gamma$ if there is a sequence $\beta_1, \ldots, \beta_n$ of atoms such that:
Let $\alpha$ and $\gamma$ be two distinct atoms of $Q$.

We say that $\alpha$ attacks $\gamma$ if there is a sequence $\beta_1, \ldots, \beta_n$ of atoms such that:

- $\alpha = \beta_1$ and $\beta_n = \gamma$
Let $\alpha$ and $\gamma$ be two distinct atoms of $Q$

- We say that $\alpha$ attacks $\gamma$ if there is a sequence $\beta_1, \ldots, \beta_n$ of atoms such that:
  - $\alpha = \beta_1$ and $\beta_n = \gamma$
  - Every $\text{Var}(\beta_i) \cap \text{Var}(\beta_{i+1})$ contains at least one variable that is externally independent of $\alpha$
If $\beta$ and $\gamma$ are atoms, then we denote by $\beta \overset{x}{\sim}_R \gamma$ the fact that $x$ is a variable in $\text{Var}(\beta) \cap \text{Var}(\gamma)$ that is externally independent of $\alpha_R$. 
If $\beta$ and $\gamma$ are atoms, then we denote by $\beta \xrightarrow{x} R \gamma$ the fact that $x$ is a variable in $\text{Var}(\beta) \cap \text{Var}(\gamma)$ that is externally independent of $\alpha_R$.

Hence, $\alpha$ attacks $\gamma$ if and only if there exists a sequence

$$\beta_1 \xrightarrow{x_1} R \beta_2 \xrightarrow{x_2} R \cdots \xrightarrow{x_{n-1}} R \beta_n$$

where $\beta_1 = \alpha$, $R = R_\alpha$, and $\beta_n = \gamma$.
\[ Q() :\sim R(x,a,y), S(x,z,w), T(z,y,u), U(b,z) \]
Examples

\[ Q() \equiv \overline{R(x, a, y)} \land \overline{S(x, z, w)} \land \overline{T(z, y, u)} \land \overline{U(b, z)} \]

- \( R(x, a, y) \) attacks \( T(z, y, u) \):

\[ R(x, a, y) \xrightarrow{R} T(z, y, u) \]
Examples

\[ Q() : \neg R(x, a, y), S(x, z, w), T(z, y, u), U(b, z) \]

- **\( R(x, a, y) \)** attacks **\( T(z, y, u) \):**

\[ R(x, a, y) \rightarrow_R T(z, y, u) \]

- **\( U(b, z) \)** attacks all other atoms:

\[ U(b, z) \rightarrow_U S(x, z, w) \rightarrow_U R(x, a, y) \rightarrow_U T(z, y, u) \]
If $\alpha$ attacks $\gamma$ then we say that:

- $\alpha$ weakly attacks $\gamma$ if $\text{FD}(Q)$ implies $\text{KVar}(\alpha) \rightarrow \text{KVar}(\gamma)$
- $\alpha$ strongly attacks $\gamma$ otherwise
Weak and Strong Attack

- If $\alpha$ attacks $\gamma$ then we say that:
  - $\alpha$ weakly attacks $\gamma$ if $\text{FD}(Q)$ implies $\text{KVar}(\alpha) \rightarrow \text{KVar}(\gamma)$
  - $\alpha$ strongly attacks $\gamma$ otherwise

- Example: $Q() :- R(x, a, y), S(x, z, w), T(z, y, u), U(b, z)$
Weak and Strong Attack

- If $\alpha$ attacks $\gamma$ then we say that:
  - $\alpha$ weakly attacks $\gamma$ if $\text{FD}(Q)$ implies $\text{KVar}(\alpha) \rightarrow \text{KVar}(\gamma)$
  - $\alpha$ strongly attacks $\gamma$ otherwise

- Example: $Q() :\neg R(x, a, y), S(x, z, w), T(z, y, u), U(b, z)$
  - $R(x, a, y)$ weakly attacks $T(z, y, u)$
Weak and Strong Attack

- If $\alpha$ attacks $\gamma$ then we say that:
  - $\alpha$ weakly attacks $\gamma$ if $\text{FD}(Q)$ implies $\text{KVar}(\alpha) \rightarrow \text{KVar}(\gamma)$
  - $\alpha$ strongly attacks $\gamma$ otherwise

- Example: $Q() :- R(x, a, y), S(x, z, w), T(z, y, u), U(b, z)$
  - $R(x, a, y)$ weakly attacks $T(z, y, u)$
  - $U(b, z)$ strongly attacks all other atoms
The attack graph of $Q$ is the directed graph $G = (V, E)$ where:

- $V$ is $\text{Atoms}(Q)$
- There is an edge $(\alpha, \gamma)$ whenever $\alpha$ attacks $\gamma$
The attack graph of $Q$ is the directed graph $G = (V, E)$ where:

- $V$ is $\text{Atoms}(Q)$
- There is an edge $(\alpha, \gamma)$ whenever $\alpha$ attacks $\gamma$

An edge $(\alpha, \gamma)$ is:

- weak if $\alpha$ weakly attacks $\gamma$ (i.e., $\text{FD}(Q)$ implies $\text{KVar}(\alpha) \rightarrow \text{KVar}(\gamma)$)
- strong if $\alpha$ strongly attacks $\gamma$
\[ Q() :- R(x, a, y), S(x, z, w), T(z, y, u), U(b, z) \]
Theorem [KW15]

Let $S = (\mathcal{R}, \Sigma)$ be a schema, such that $\Sigma$ consists of primary keys. Let $Q$ be a Boolean CQ without self joins, and let $G$ be the attack graph of $Q$. 

1. If $G$ is acyclic, then $\text{Consistent}_S Q$ is expressible in FO.
2. If $G$ has cycles, but no cycle contains a strong edge, then $\text{Consistent}_S Q$ can be computed in polynomial time.
3. If $G$ has a cycle with a strong edge, then it is coNP-complete to decide whether $\text{Consistent}_S Q$ is true on a given instance.
Refined Trichotomy (Boolean case)

**Theorem [KW15]**

Let $S = (R, \Sigma)$ be a schema, such that $\Sigma$ consists of primary keys. Let $Q$ be a Boolean CQ without self joins, and let $G$ be the attack graph of $Q$.

1. $G$ is acyclic if and only if $\text{Consistent}_Q^\Sigma$ is expressible in FO.
Refined Trichotomy (Boolean case)

**Theorem [KW15]**

Let $S = (R, \Sigma)$ be a schema, such that $\Sigma$ consists of primary keys. Let $Q$ be a Boolean CQ without self joins, and let $G$ be the attack graph of $Q$.

1. $G$ is acyclic if and only if $\text{Consistent}^Q_\Sigma$ is expressible in FO.
2. If $G$ has cycles, but no cycle contains a strong edge, then $\text{Consistent}^Q_\Sigma$ can be computed in polynomial time.
Theorem [KW15]

Let $S = (\mathcal{R}, \Sigma)$ be a schema, such that $\Sigma$ consists of primary keys. Let $Q$ be a Boolean CQ without self joins, and let $G$ be the attack graph of $Q$.

1. $G$ is acyclic if and only if $\text{Consistent}^Q_\Sigma$ is expressible in FO.
2. If $G$ has cycles, but no cycle contains a strong edge, then $\text{Consistent}^Q_\Sigma$ can be computed in polynomial time.
3. If $G$ has a cycle with a strong edge, then it is coNP-complete to decide whether $\text{Consistent}^Q_\Sigma$ is true on a given instance.
Example 1

\[ Q() \equiv R(x, a, y), S(x, z, w), T(z, y, u), U(b, z) \]
Example 1

\[
Q() : - R(x, a, y), S(x, z, w), T(z, y, u), U(b, z)
\]

⇒ in FO
Example 2

<table>
<thead>
<tr>
<th>LC(lecturer, course)</th>
<th>CT(course, ta)</th>
</tr>
</thead>
<tbody>
<tr>
<td>lecturer (\rightarrow) course</td>
<td>course (\rightarrow) ta</td>
</tr>
</tbody>
</table>
Example 2

<table>
<thead>
<tr>
<th>LC(lecturer, course)</th>
<th>CT(course, ta)</th>
</tr>
</thead>
<tbody>
<tr>
<td>lecturer → course</td>
<td>course → ta</td>
</tr>
</tbody>
</table>

Query: Does any course have both a lecturer and a TA?
Example 2

\[
\begin{array}{c|c}
\text{LC(lecturer,course)} & \text{CT(course,ta)} \\
\text{lecturer} \rightarrow \text{course} & \text{course} \rightarrow \text{ta} \\
\end{array}
\]

Query: Does any course have both a lecturer and a TA?

\[
Q() :\neg \text{LC}(x,y), \text{CT}(y,z)
\]
Example 2

<table>
<thead>
<tr>
<th>LC(lecturer,course)</th>
<th>CT(course,ta)</th>
</tr>
</thead>
<tbody>
<tr>
<td>lecturer → course</td>
<td>course → ta</td>
</tr>
</tbody>
</table>

Query: Does any course have both a lecturer and a TA?

\[ Q() :\neg \text{LC}(x, y), \text{CT}(y, z) \]
Example 2

\[
\begin{array}{ll}
\text{LC(lecturer, course)} & \text{CT(course, ta)} \\
\text{lecturer} \rightarrow \text{course} & \text{course} \rightarrow \text{ta}
\end{array}
\]

Query: Does any course have both a lecturer and a TA?

\[Q() \dashv \text{LC}(x, y), \text{CT}(y, z)\]

\[
\begin{array}{c}
\text{LC}(x, y) \\
\text{CT}(y, z)
\end{array}
\]

\[\Rightarrow \text{in FO}\]
Example 3

<table>
<thead>
<tr>
<th>LC(lecturer, course)</th>
<th>TC(ta, course)</th>
</tr>
</thead>
<tbody>
<tr>
<td>lecturer → course</td>
<td>ta → course</td>
</tr>
</tbody>
</table>

Query: Does any course have both a lecturer and a TA?
Example 3

<table>
<thead>
<tr>
<th>LC(lecturer,course)</th>
<th>TC(ta,course)</th>
</tr>
</thead>
<tbody>
<tr>
<td>lecturer → course</td>
<td>ta → course</td>
</tr>
</tbody>
</table>

Query: Does any course have both a lecturer and a TA?

\[ Q() : - \text{LC}(x, y), \text{TC}(x', y) \]
Example 3

The query is to determine if any course has both a lecturer and a TA. The query can be expressed as:

\[ Q() \rightarrow \text{LC}(x, y), \text{TC}(x', y) \]

This query is coNP-complete. The table below summarizes the rules for the example:

<table>
<thead>
<tr>
<th>Lecture (LC)</th>
<th>TA (TC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lecturer → Course</td>
<td>TA → Course</td>
</tr>
</tbody>
</table>

Query: Does any course have both a lecturer and a TA?
Example 3

\[
\begin{array}{c|c}
\text{LC(lecturer,course)} & \text{TC(ta,course)} \\
\text{lecturer} \rightarrow \text{course} & \text{ta} \rightarrow \text{course}
\end{array}
\]

Query: Does any course have both a lecturer and a TA?

\[Q() :\neg \text{LC}(x,y), \text{TC}(x',y)\]

\[
\text{LC}(x,y) \leftrightarrow \text{TC}(x',y)
\]

\[\Rightarrow \text{coNP-complete}\]
Example 4

\[
\begin{array}{c|c}
\text{LC(lecturer,course)} & \text{CT(course,ta)} \\
\text{lecturer} \rightarrow \text{course} & \text{course} \rightarrow \text{ta}
\end{array}
\]

Query: Does any course have the same lecturer and TA?
Example 4

<table>
<thead>
<tr>
<th>LC(lecturer, course)</th>
<th>CT(course, ta)</th>
</tr>
</thead>
<tbody>
<tr>
<td>lecturer → course</td>
<td>course → ta</td>
</tr>
</tbody>
</table>

Query: Does any course have the same lecturer and TA?

\[ Q() \leftarrow LC(x, y), CT(y, x) \]

\[ LC(x, y) \leftrightarrow CT(y, x) \]
Example 4

<table>
<thead>
<tr>
<th>LC(lecturer, course)</th>
<th>CT(course, ta)</th>
</tr>
</thead>
<tbody>
<tr>
<td>lecturer → course</td>
<td>course → ta</td>
</tr>
</tbody>
</table>

Query: Does any course have the same lecturer and TA?

\[
Q() \colon \neg \text{LC}(x, y), \text{CT}(y, x)
\]

⇒ not in FO, but in polynomial time
Proof of Polynomial Time

- The proof of the non-FO polynomial time is the most involved in the proof of the trichotomy.
The proof of the non-FO polynomial time is the most involved in the proof of the trichotomy.

We will see the proof of Kolaitis and Pema [KP12] for the CQ

\[ Q() \rightarrow \text{LC}(x,y), \text{CT}(y,x) \]
Conflict-Join Graph

- $Q() :− \text{LC}(x, y), \text{CT}(y, x)$
- For an instance $I$, the conflict-join graph of $I$, denoted $G_{Q,I}$, is the undirected graph with the following properties:
Conflict-Join Graph

- $Q() \leftarrow \text{LC}(x, y), \text{CT}(y, x)$
- For an instance $I$, the conflict-join graph of $I$, denoted $G_{Q,I}$, is the undirected graph with the following properties:
  - The nodes are all the facts $\text{LC}(a, b)$ and $\text{CT}(c, d)$ of $I$
Conflict-Join Graph

- $Q() \leftarrow \text{LC}(x, y), \text{CT}(y, x)$

- For an instance $I$, the \textit{conflict-join} graph of $I$, denoted $G_{Q,I}$, is the undirected graph with the following properties:
  - The nodes are all the facts $\text{LC}(a, b)$ and $\text{CT}(c, d)$ of $I$
  - There is an edge between:
Conflict-Join Graph

- $Q() := \text{LC}(x, y), \text{CT}(y, x)$
- For an instance $I$, the conflict-join graph of $I$, denoted $G_{Q,I}$, is the undirected graph with the following properties:
  - The nodes are all the facts $\text{LC}(a, b)$ and $\text{CT}(c, d)$ of $I$
  - There is an edge between:
    - every two conflicting facts $\text{LC}(a, b)$ and $\text{LC}(a, b')$
Conflict-Join Graph

- \( Q() \) := \( LC(x, y), CT(y, x) \)
- For an instance \( I \), the conflict-join graph of \( I \), denoted \( G_{Q,I} \), is the undirected graph with the following properties:
  - The nodes are all the facts \( LC(a, b) \) and \( CT(c, d) \) of \( I \)
  - There is an edge between:
    - every two conflicting facts \( LC(a, b) \) and \( LC(a, b') \)
    - every two conflicting facts \( CT(c, d) \) and \( CT(c, d') \)
For an instance $I$, the conflict-join graph of $I$, denoted $G_{Q,I}$, is the undirected graph with the following properties:

- The nodes are all the facts $\text{LC}(a,b)$ and $\text{CT}(c,d)$ of $I$.
- There is an edge between:
  - every two conflicting facts $\text{LC}(a,b)$ and $\text{LC}(a,b')$.
  - every two conflicting facts $\text{CT}(c,d)$ and $\text{CT}(c,d')$.
  - every two joinable facts $\text{LC}(a,b)$ and $\text{CT}(b,a)$. 

\[
Q() : \neg \text{LC}(x,y), \text{CT}(y,x).
\]
Example of a Conflict-Join Graph $G_{Q,I}$

```
CT(PL, Keren)          CT(PL, Eran)
/\                     /\                      \
LC(Keren, PL)          LC(Eran, PL)               LC(Eran, AI)
/\                     /\                      \
LC(Keren, AI)          LC(Eran, AI)               LC(Eran, DB)
/\                     /\                      \
LC(Keren, DB)          LC(Eran, DB)               CT(AI, Eran)
/\                     /\                      \
CT(DB, Keren)          CT(AI, Eran)               
```
Lemma

**Lemma [KP12]**

Consider the CQ $Q() \colon \neg \text{LC}(x, y), \text{CT}(y, x)$ and an inconsistent instance $I$. Let $n$ be the number of keys (in the two relations) in $I$. The following are equivalent:

- There exists a repair $J$ of $I$ with $Q(J) = \text{false}$
- $G_{Q,I}$ has an independent set of size $n$
Example of an Independent Set of $G_{Q,I}$
Problem?

- Determining whether a graph has an independent set of a given size is **NP-complete**!
  - So how does the lemma help us?
Determining whether a graph has an independent set of a given size is **NP-complete**!

- *So how does the lemma help us?*

But for some types of graphs, this problem is known to be solvable in polynomial time; for example:

- Chordal graphs
- Perfect graphs
- Graphs with a bounded treewidth
- **Claw-free** graphs (Minty [Min80])
Problem?

- Determining whether a graph has an independent set of a
given size is **NP-complete!**
  - *So how does the lemma help us?*
- But for some types of graphs, this problem is known to be
  solvable in polynomial time; for example:
  - Chordal graphs
  - Perfect graphs
  - Graphs with a bounded treewidth
  - **Claw-free** graphs (Minty [Min80])

- A **claw** is the complete bipartite graph $K_{1,3}$
Problem?

- Determining whether a graph has an independent set of a given size is **NP-complete**!
  - *So how does the lemma help us?*
- But for some types of graphs, this problem is known to be solvable in polynomial time; for example:
  - Chordal graphs
  - Perfect graphs
  - Graphs with a bounded treewidth
  - **Claw-free** graphs (Minty [Min80])

- A **claw** is the complete bipartite graph $K_{1,3}$
- A graph $g$ is **claw free** if no **induced** subgraph of $g$ is a claw
Can You Find an Induced Claw?

CT(PL, Keren)  CT(PL, Eran)

<table>
<thead>
<tr>
<th>LC(Keren, PL)</th>
<th>LC(Eran, PL)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LC(Keren, AI)</td>
<td>LC(Eran, AI)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LC(Keren, DB)</th>
<th>LC(Eran, DB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CT(DB, Keren)</td>
<td></td>
</tr>
</tbody>
</table>

CT(AI, Eran)
Is this an Induced Claw?

CT(PL, Keren)  CT(PL, Eran)

LC(Keren, PL)  LC(Eran, PL)

LC(Keren, AI)  LC(Eran, AI)

LC(Keren, DB)  LC(Eran, DB)

CT(DB, Keren)  CT(AI, Eran)
Completing the Proof

- Lemma: \( G_{Q,I} \) is claw free.
Completing the Proof

- Lemma: $G_{Q,I}$ is claw free.
- Corollary: for $Q() :\neg \text{LC}(x,y), \text{CT}(y,x)$ the consistency problem can be solved in polynomial time
To extend the trichotomy to non-Boolean CQs, we need some notation.
Extending to Non-Boolean CQs

- To extend the trichotomy to non-Boolean CQs, we need some notation.
- If \( x \) is a sequence of variables and \( a \) is a sequence of constants of the same length as \( x \), then \( Q[x \rightarrow a] \) is the CQ that is obtained from \( Q \) by replacing each variable \( x_i \) with \( a_i \).
To extend the trichotomy to non-Boolean CQs, we need some notation.

- If \( x \) is a sequence of variables and \( a \) is a sequence of constants of the same length as \( x \), then \( Q[x\rightarrow a] \) is the CQ that is obtained from \( Q \) by replacing each variable \( x_i \) with \( a_i \).
  - If \( x_i \) is a head variable, then we remove \( x_i \) from the head.
Generalized Trichotomy (non-Boolean case)

**Theorem [KW15]**

Let $S = (\mathcal{R}, \Sigma)$ be a schema, such that $\Sigma$ consists of primary keys. Let $Q(x)$ be a CQ without self joins (where $x$ is the sequence of head variables). Let $Q_b$ be the Boolean CQ $Q[x \rightarrow a]$ for some tuple $a$ of constants, and let be the attack graph of $Q_b$. 

1. $G$ is acyclic if and only if $\text{Consistent } Q[\Sigma]$ is equivalent to some $\varphi(x)$ in FO.
2. If $G$ has cycles, but no cycle contains a strong edge, then $\text{Consistent } Q[\Sigma]$ can be evaluated in polynomial time.
3. If $G$ has a cycle with a strong edge, then non-emptiness of $\text{Consistent } Q[\Sigma]$ is coNP-complete.
Generalized Trichotomy (non-Boolean case)

**Theorem [KW15]**

Let $\mathcal{S} = (\mathcal{R}, \Sigma)$ be a schema, such that $\Sigma$ consists of primary keys. Let $Q(x)$ be a CQ without self joins (where $x$ is the sequence of head variables). Let $Q_b$ be the Boolean CQ $Q[x \rightarrow a]$ for some tuple $a$ of constants, and let $G$ be the attack graph of $Q_b$.

1. $G$ is acyclic if and only if $\text{Consistent}^Q_\Sigma$ is equivalent to some $\varphi(x)$ in FO.
Generalized Trichotomy (non-Boolean case)

**Theorem [KW15]**

Let \( S = (\mathcal{R}, \Sigma) \) be a schema, such that \( \Sigma \) consists of primary keys. Let \( Q(x) \) be a CQ without self joins (where \( x \) is the sequence of head variables). Let \( Q_b \) be the Boolean CQ \( Q[x \rightarrow a] \) for some tuple \( a \) of constants, and let be the attack graph of \( Q_b \).

1. \( G \) is acyclic if and only if \( \text{Consistent}^Q_{\Sigma} \) is equivalent to some \( \varphi(x) \) in FO.

2. If \( G \) has cycles, but no cycle contains a strong edge, then \( \text{Consistent}^Q_{\Sigma} \) can be evaluated in polynomial time.
**Theorem [KW15]**

Let $S = (\mathcal{R}, \Sigma)$ be a schema, such that $\Sigma$ consists of primary keys. Let $Q(x)$ be a CQ without self joins (where $x$ is the sequence of head variables). Let $Q_b$ be the Boolean CQ $Q[x \rightarrow a]$ for some tuple $a$ of constants, and let be the attack graph of $Q_b$.

1. $G$ is acyclic if and only if $\text{Consistent}_{\Sigma}^Q$ is equivalent to some $\varphi(x)$ in FO.

2. If $G$ has cycles, but no cycle contains a strong edge, then $\text{Consistent}_{\Sigma}^Q$ can be evaluated in polynomial time.

3. If $G$ has a cycle with a strong edge, then non-emptiness of $\text{Consistent}_{\Sigma}^Q$ is coNP-complete.
Example

\[ Q(y) :\neg R(x, a, y), S(x, z, w), T(z, y, u), U(b, z) \]
\[ \downarrow \]
\[ Q'(\cdot) :\neg R(x, a, c), S(x, z, w), T(z, c, u), U(b, z) \]

\begin{tikzpicture}
    \node (R) at (0,0) {$R(x, a, c)$};
    \node (U) at (2,0) {$U(b, z)$};
    \node (T) at (0,-2) {$T(z, c, u)$};
    \node (S) at (2,-2) {$S(x, z, w)$};
    \draw[->] (R) -- (U);\draw[dotted] (T) -- (S);
\end{tikzpicture}
Example

\[ Q(y) :- R(x, a, y), S(x, z, w), T(z, y, u), U(b, z) \]
\[ \downarrow \]
\[ Q'(x) :- R(x, a, c), S(x, z, w), T(z, c, u), U(b, z) \]

\[ \Rightarrow \text{in FO} \]
Let $\mathcal{S} = (\mathcal{R}, \Sigma)$ be a schema, such that $\Sigma$ consists of primary keys (one per relation name).
Problem Definition

- Let $\mathcal{S} = (\mathcal{R}, \Sigma)$ be a schema, such that $\Sigma$ consists of primary keys (one per relation name)
- Given: a CQ $Q$ over $\mathcal{R}$ such that
  - $Q$ has no self joins (i.e., no relation name occurs more than once)
  - The attack graph of $Q$ is acyclic
Problem Definition

- Let $S = (\mathcal{R}, \Sigma)$ be a schema, such that $\Sigma$ consists of primary keys (one per relation name)
- Given: a CQ $Q$ over $\mathcal{R}$ such that
  - $Q$ has no self joins (i.e., no relation name occurs more than once)
  - The attack graph of $Q$ is acyclic
- Goal: Compute an SQL query $Q_{cqa}$ over $\mathcal{R}$, such that for every inconsistent instance $I$ we have:

$$\text{Consistent}_\Sigma^Q(I) = Q_{cqa}(I)$$
We denote $\alpha \in \text{Atoms}(Q)$ by $\alpha(x, y)$, where $x$ is $\text{KVar}(\alpha)$ and $y$ is $\text{Var}(\alpha) \setminus \text{KVar}(\alpha)$. 
Notation

- We denote $\alpha \in \text{Atoms}(Q)$ by $\alpha(x, y)$, where $x$ is $\text{KVar}(\alpha)$ and $y$ is $\text{Var}(\alpha) \setminus \text{KVar}(\alpha)$.

- If $\alpha \in \text{Atoms}(Q)$, then $Q^{-\alpha}$ is the CQ obtained from $Q$ by removing $\alpha$. 
We denote $\alpha \in \text{Atoms}(Q)$ by $\alpha(x, y)$, where $x$ is $\text{KVar}(\alpha)$ and $y$ is $\text{Var}(\alpha) \setminus \text{KVar}(\alpha)$.

If $\alpha \in \text{Atoms}(Q)$, then $Q^{-\alpha}$ is the CQ obtained from $Q$ by removing $\alpha$.

Recall: if $x$ is a sequence of variables and $a$ is a sequence of constants of the same length as $x$, then $Q[x \rightarrow a]$ is the CQ that is obtained from $Q$ by replacing each variable $x_i$ with $a_i$. 
We denote $\alpha \in \text{Atoms}(Q)$ by $\alpha(x, y)$, where $x$ is $\text{KVar}(\alpha)$ and $y$ is $\text{Var}(\alpha) \setminus \text{KVar}(\alpha)$.

If $\alpha \in \text{Atoms}(Q)$, then $Q^{-\alpha}$ is the CQ obtained from $Q$ by removing $\alpha$.

Recall: if $x$ is a sequence of variables and $a$ is a sequence of constants of the same length as $x$, then $Q[x \rightarrow a]$ is the CQ that is obtained from $Q$ by replacing each variable $x_i$ with $a_i$.

It may be the case that $Q$ does not contain some of the $x_i$. 
**Key Lemma**

**Lemma [KW15]**

Let $S = (R, \Sigma)$ be a schema where $\Sigma$ consists of primary keys. Let $Q$ be a Boolean CQ without self joins, and $I$ an inconsistent instance. Let $\alpha(x, y)$ be an atom without incoming edges in the attack graph of $Q$. The following are equivalent.

1. $Q$ is consistent (i.e., true on every repair of) over $I$.
2. For some $\alpha(a, b) \in I$, the CQ $Q[x \rightarrow a]$ is consistent over $I$. 
3. There is a fact $f = \alpha(a, b) \in I$ such that: for all facts $g$ of $R$ $\alpha$ with the key of $f$ there is $c$ such that: (1) $g = \alpha(a, c)$, and (2) $Q - \alpha[(x, y) \rightarrow (a, c)]$ is consistent over $I$. 

Lemma [KW15]

Let $S = (\mathcal{R}, \Sigma)$ be a schema where $\Sigma$ consists of primary keys. Let $Q$ be a Boolean CQ without self joins, and $I$ an inconsistent instance. Let $\alpha(x, y)$ be an atom without incoming edges in the attack graph of $Q$. The following are equivalent.

- $Q$ is consistent (i.e., true on every repair of) over $I$. 

**Key Lemma**

**Lemma [KW15]**

Let $S = (\mathcal{R}, \Sigma)$ be a schema where $\Sigma$ consists of primary keys. Let $Q$ be a Boolean CQ without self joins, and $I$ an inconsistent instance. Let $\alpha(x, y)$ be an atom without incoming edges in the attack graph of $Q$. The following are equivalent.

- $Q$ is consistent (i.e., true on every repair of) over $I$.

- For some $\alpha(a, b) \in I$, the CQ $Q[x \rightarrow a]$ is consistent over $I$. 
**Key Lemma**

**Lemma [KW15]**

Let \( S = (R, \Sigma) \) be a schema where \( \Sigma \) consists of primary keys. Let \( Q \) be a Boolean CQ without self joins, and \( I \) an inconsistent instance. Let \( \alpha(x, y) \) be an atom without incoming edges in the attack graph of \( Q \). The following are equivalent.

- \( Q \) is consistent (i.e., true on every repair of) over \( I \).
- For some \( \alpha(a, b) \in I \), the CQ \( Q_{[x \to a]} \) is consistent over \( I \).
- There is a fact \( f = \alpha(a, b) \in I \) such that: for all facts \( g \) of \( R_\alpha \) with the key of \( f \) there is \( c \) such that: (1) \( g = \alpha(a, c) \), and (2) \( Q_{[(x, y) \to (a, c)]} \) is consistent over \( I \).
Another Lemma

**Lemma [KW15]**

Let \( S = (R, \Sigma) \) be a schema where \( \Sigma \) consists of primary keys. Let \( Q \) be a Boolean CQ without self joins and with an acyclic attack graph. Let \( \alpha(x, y) \) be an atom without incoming edges in the attack graph of \( Q \).
**Lemma [KW15]**

Let $S = (\mathcal{R}, \Sigma)$ be a schema where $\Sigma$ consists of primary keys. Let $Q$ be a Boolean CQ without self joins and with an acyclic attack graph. Let $\alpha(x, y)$ be an atom without incoming edges in the attack graph of $Q$. For every $\alpha(a, c) \in I$, the CQ $Q^-\alpha[\langle x, y \rangle \rightarrow (a, c)]$ has an acyclic attack graph.
We denote $Q$ as the following SQL query:

```
SELECT X FROM R WHERE AC AND TC
```

Where:

- $R$ is a sequence $R_1, \ldots, R_m$ of relation names
- $X$ is a sequence of variables of the form $R_i.A$
  - $A$ is an attribute of $R$
- $AC$ is a conjunction of conditions of the form $R_i.A = R_j.B$
- $TC$ is a conjunction of conditions of the form $R_i.A = t$ where $t$ is some term (initially a constant)
Notation

- For a counter $l$, we denote by
  - $R^l$ the sequence obtained from $R$ by replacing each $R_i$ with 
    “$R_i R_i^l$” (i.e., naming $R_i$ by $R_i^l$)
  - $X^l$ the sequence obtained from $X$ by replacing each $R_i.A$ with 
    $R_i^l.A$
  - $AC^l$ the conjunction obtained from $AC$ by replacing each 
    $R_i.A = R_j.B$ with $R_i^l.A = R_j^l.B$
  - $TC^l$ the conjunction obtained from $TC$ by replacing each 
    $R_i.A = t$ with $R_i^l.A = t$
If \( R' \) is a subsequence of \( R \), then we denote by

- \( \text{AC} \cap R' \) the restriction of \( \text{AC} \) to those \( R_i.A = R_j.B \) where \( R_i \in R' \) and \( R_j \in R' \)
- \( \text{TC} \cap R' \) the restriction of \( \text{AC} \) to those \( R_i.A = t \) where \( R_i \in R' \)
Selecting a Non-Attacked Atom

- Let $\alpha$ be a non-attacked atom (i.e., $\alpha$ has no incoming edges in the attack graph), and let $R = R_\alpha$
Selecting a Non-Attacked Atom

- Let \( \alpha \) be a non-attacked atom (i.e., \( \alpha \) has no incoming edges in the attack graph), and let \( R = R_{\alpha} \)
- Denote by:
  - \( K = R.A_1, \ldots, R.A_k \) the key attributes of \( R \)
  - \( V = R.B_1, \ldots, R.B_q \) the non-key attributes of \( R_i \)
Recursive Rewriting

- Begin with Boolean: assuming 'true' instead of X

```
SELECT 'true' FROM R WHERE AC AND TC
```
Recursive Rewriting

- Begin with Boolean: assuming 'true' instead of $X$

\[
\text{SELECT 'true' FROM } R \text{ WHERE } AC \text{ AND } TC
\]

- We create the rewriting $\text{Rewrite}(R, AC, TC)$:

\[
\text{SELECT 'true' FROM } R \ R^1 \text{ WHERE NOT EXISTS (}
\text{SELECT 'true' FROM } R \ R^2 \text{ WHERE } K^2 = K^1 \text{ AND NOT (}
\text{ ( } AC^2 \cap \{ R^2 \} \text{ AND } TC^2 \cap \{ R^2 \} \text{ ) AND NOT (}
\text{EXISTS(} \text{Rewrite}(R', AC \cap R', TC') \text{))})
\]
Recursive Rewriting

- Begin with Boolean: assuming 'true' instead of $X$

  ```sql
  SELECT 'true' FROM R WHERE AC AND TC
  ```

- We create the rewriting $\text{Rewrite}(R, AC, TC)$:

  ```sql
  SELECT 'true' FROM R R¹ WHERE 
  NOT EXISTS (
    SELECT 'true' FROM R R² WHERE K² = K¹ AND NOT 
    ( ( AC² ∩ \{R²\} AND TC² ∩ \{R²\} ) AND 
    EXISTS(\text{Rewrite}(R', AC ∩ R', TC')) )
  )
  ```

- $R'$ is obtained from $R$ by removing $R$

- $TC'$ is obtained from $TC ∩ R'$ by adding $R_k.A = R².B$ for every condition in $AC$ of the form $R_k.A = R.B$ or $R.B = R_k.A$ where $R_k \neq R$
Example 1

\[
LC(Ax, Ay) ; CT(Ay, Az) \quad Q() :\neg LC(x, y), CT(y, z)
\]

\[
\begin{array}{c}
LC(x, y) \\
\end{array} \rightarrow 
\begin{array}{c}
CT(y, z)
\end{array}
\]

\[
\text{SELECT } 'true' \text{ FROM } LC, CT \text{ WHERE } LC.Ay=CT.Ay
\]

\[
\begin{array}{c}
\text{AC}
\end{array}
\]
Example 1

\[
\text{SELECT 'true' FROM LC, CT WHERE LC.Ay=CT.Ay}
\]

\[
\text{SELECT 'true' FROM LC LC1 WHERE NOT EXISTS (}
\text{SELECT 'true' FROM LC LC2 WHERE LC2.Ax=LC1.Ax AND NOT (}
\text{EXISTS (}
\text{SELECT 'true' FROM CT CT1 WHERE NOT EXISTS (}
\text{SELECT 'true' FROM CT CT2 WHERE CT2.Ay=CT1.Ay AND NOT ( CT2.Ay = LC2.Ay )}
\text{))}
\text{))}
\]
Example 1

SELECT 'true' FROM LC, CT WHERE LC.Ay = CT.Ay

SELECT 'true' FROM LC LC1 WHERE NOT EXISTS (
  SELECT 'true' FROM LC LC2 WHERE LC2.Ax = LC1.Ax AND NOT
  EXISTS(
    SELECT 'true' FROM CT WHERE CT.Ay = LC2.Ay
  )
)
Example 2

\[ \text{LC}(Ax, Ay) ; \text{CT}(Ay, Az) \quad Q() := \text{LC}(x, y), \text{CT}(y, Avi) \]

\[ \text{SELECT 'true' FROM LC, CT WHERE LC.Ay=CT.Ay AND CT.Az='Avi'} \]

\[ \text{AC} \quad \text{TC} \]
Example 2

\[
\text{SELECT 'true' FROM LC, CT WHERE LC.Ay=CT.Ay AND CT.Az='Avi'}
\]

\[
\text{SELECT 'true' FROM LC \_LC1 WHERE NOT EXISTS (}
\]
\[
\text{SELECT 'true' FROM LC \_LC2 WHERE LC2.Ax=LC1.Ax AND NOT (}
\]
\[
\text{EXISTS (}
\]
\[
\text{SELECT 'true' FROM CT \_CT1 WHERE NOT EXISTS (}
\]
\[
\text{SELECT 'true' FROM CT \_CT2 WHERE}
\]
\[
\]
\[
\text{))}
\]
Non-Boolean Case

\[
\text{SELECT } X \text{ FROM } R \text{ WHERE } AC \text{ AND } TC
\]
Non-Boolean Case

\[
\text{SELECT } X \text{ FROM } R \text{ WHERE } AC \text{ AND } TC
\]

⇓

\[
\text{SELECT } X^0 \text{ FROM } R^0 \text{ WHERE EXISTS (Rewrite}(R, AC, TC'))
\]

\(TC'\) is obtained from \(TC\) by adding \(R_i.A = R^0_i.A\) for every \(R_i.A\) in \(X\)
Example 3

\[ LC(Ax, Ay); CT(Ay, Az) \quad Q(z) :\neg LC(x, y), CT(y, z) \]

\[ \text{SELECT CT.Az FROM LC, CT WHERE LC.Ay} = \text{CT.Ay} \]

\[ AC \]
Example 3

SELECT CT.Az FROM LC, CT WHERE LC.Ay=CT.Ay

\[
\text{AC}
\]

SELECT CT0.Az FROM LC LC0, CT CT0 WHERE EXISTS(
    SELECT 'true' FROM LC LC1 WHERE NOT EXISTS ( 
        SELECT 'true' FROM LC LC2 WHERE LC2.Ax=LC1.Ax AND NOT EXISTS(
            SELECT 'true' FROM CT CT1 WHERE NOT EXISTS ( 
                SELECT 'true' FROM CT CT2 WHERE 
                    CT2.Ay=CT1.Ay AND NOT
                    ( CT2.Ay = LC2.Ay AND CT2.Az = CT0.Az)
            )))
    )
)


References II


End of lecture 8

Consistent Query Answering