Principles of Managing Uncertain Data

Lecture 7: Inconsistent Databases
Table of Contents

1. Introduction
2. Inconsistent Databases
3. Repairs
4. Consistent Query Answering
5. Complexity Aspects
## Table of Contents

1. Introduction
2. Inconsistent Databases
3. Repairs
4. Consistent Query Answering
5. Complexity Aspects
Inconsistency in Databases

- Various applications rely on inconsistent data:
  - Multiple, autonomous sources of data
    - Each may be consistent, but there may be disagreements across different sources
  - Data with potential errors (e.g., socially-maintained encyclopedias)
  - Imprecise data-generation processes (e.g., text extraction)
- In a database context, inconsistency means that we have *integrity constraints* (phrased over the schema), and these are *violated*
So, What to Do?

- Manual correction of data
  - Very limited in scale, not always possible
- Heuristic automated *cleaning* (e.g., if a person has two salaries, take the average)
  - Very common approach
  - Valuable information may be lost
  - Significant errors may be introduced
- **Consistent query answering**
  - Do the best you can without resolving conflicts
  - *This lecture*
Consistent Query Answering (CQA)

- Introduced in 1999 by Arenas, Bertossi Chomicki [ABC99]
- Idea: query engine considers all possible ways of “repairing” the data
  - A repair should mimic a legitimate manual cleaning
  - Formal definitions always involve a notion of a “minimal change”
- To answer a query, give the answers that are valid no matter which repair is being used
- Ideally, considering “all possible repairs” is only conceptual, and efficient algorithms answer queries much more efficiently
  - As we shall see, some combinations of queries and constraints allow for efficiency; others do not
Research on CQA

- Lots of research followed the 1999 paper [ABC99]
  - > 830 citations currently in Google Scholar
- Complexity and algorithmic approaches to CQA
- Different classes of queries and integrity constraints
- Richer/different notions of “repairs”
  - Different update actions
  - Different notions of minimality
  - Tuple preferences to refine the cleaning process
    - Research @Technion
Table of Contents

1 Introduction

2 Inconsistent Databases

3 Repairs

4 Consistent Query Answering

5 Complexity Aspects
Recalling Schemas

- A schema $\mathcal{S}$ is a pair $(\mathcal{R}, \Sigma)$, where $\mathcal{R} = \{R_1, \ldots, R_m\}$ is a signature (set of relation schemas) and $\Sigma$ is a set of logical constraints over $\mathcal{R}$.
The framework of repairs does not restrict the kind of integrity constraints that can be used.

But the kind of constraints may have a crucial impact on the form of the repairs, as well as the complexity of CQA.
Considered Constraints

- We will focus here on two kinds of integrity constraints:
  - **Functional Dependencies (FDs)**
    - Recall: \( R : U \rightarrow V \), where \( U \) and \( V \) are sets of attributes of \( R \)
    - As a special case, we say that \( \Sigma \) consists of primary-key constraints if \( \Sigma \) associates (at most) one FD to each relation, and that FD is a key constraint
  - **Inclusion Dependencies (INDs)**
    - Recall: \( R[A_1, \ldots, A_m] \subseteq S[B_1, \ldots, B_m] \) where \( A_1, \ldots, A_m \) are distinct attributes of \( R \) and \( B_1, \ldots, B_m \) are distinct attributes of \( S \)
    - In the case of \( m = 1 \) it is called a referential constraint
Inconsistent Databases

Definition of an Inconsistent Database

Let $S = (R, \Sigma)$ be a schema. An inconsistent database is a database $I$ over $R$, such that $I$ may violate $\Sigma$. 
### Running Example

<table>
<thead>
<tr>
<th><strong>CT</strong></th>
<th></th>
<th><strong>ST</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>course</td>
<td>ta</td>
<td>student</td>
<td>track</td>
</tr>
<tr>
<td>PL</td>
<td>Ahuva</td>
<td>Ahuva</td>
<td>SWEng</td>
</tr>
<tr>
<td>OS</td>
<td>Asma</td>
<td>Asma</td>
<td>DataEng</td>
</tr>
<tr>
<td>AI</td>
<td>Avner</td>
<td>Asma</td>
<td>BioInf</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Alon</td>
<td>BioInf</td>
</tr>
</tbody>
</table>

**Constraints:**
- $CT[\text{ta}] \subseteq ST[\text{student}]$
  - This is a referential constraint
- $ST: \text{student} \rightarrow \text{track}$
  - This is an FD and a key constraint
Table of Contents

1. Introduction
2. Inconsistent Databases
3. Repairs
4. Consistent Query Answering
5. Complexity Aspects
Let $I$ and $J$ be two databases over the same signature. Recall that we view $I$ and $J$ as sets of facts $R(t)$, where $R$ is a relation and $t$ is a tuple of $R$.

The symmetric difference between $I$ and $J$, denoted $\Delta(I, J)$, is the set of all the facts in $J$ and $I$ that occur in one of the two, but not in both.

Formally, $\Delta(I, J) = (I \cup J) \setminus (I \cap J)$.

Equivalently, $\Delta(I, J) = (I \setminus J) \cup (J \setminus I)$. 

**Symmetric Difference**
Example 1

\[
\Delta(I, J) = \{\text{CT}(\text{AI}, \text{Avner}), \text{ST}(\text{Asma}, \text{BioInf})\}
\]
Example 2

$I$:

<table>
<thead>
<tr>
<th>course</th>
<th>ta</th>
</tr>
</thead>
<tbody>
<tr>
<td>PL</td>
<td>Ahuva</td>
</tr>
<tr>
<td>OS</td>
<td>Asma</td>
</tr>
<tr>
<td>AI</td>
<td>Avner</td>
</tr>
</tbody>
</table>

$J$:

<table>
<thead>
<tr>
<th>course</th>
<th>ta</th>
</tr>
</thead>
<tbody>
<tr>
<td>PL</td>
<td>Ahuva</td>
</tr>
<tr>
<td>OS</td>
<td>Asma</td>
</tr>
<tr>
<td>AI</td>
<td>Avner</td>
</tr>
</tbody>
</table>

$\Delta(I, J) = \{ST(Asma, DataEng), ST(Avner, SWEng)\}$
Let $I$ be an inconsistent database

Let $J_1$ and $J_2$ be two databases of the same signature as $I$

We say that $J_1$ is at least as close to $I$ as $J_2$, denoted $J_1 \leq_I J_2$, if $\Delta(I, J_1) \subseteq \Delta(I, J_2)$

$\leq_I$ is a partial order

- That is, reflexive, antisymmetric, and transitive

*Proof?*
**Definition (Repair) [ABC99]**

Let $S$ be a schema. Let $I$ be an inconsistent database over $S$, and let $Inst(S)$ be the set of all the (consistent) databases over $S$. A *repair* of $I$ is a member of $Inst(S)$ that is minimal under $\leq_I$. We denote by $\text{Repairs}_\Sigma(I)$ the set of all the repairs of $I$. 
### Repair Examples

Given the inconsistent database repairs $I$, we have:

<table>
<thead>
<tr>
<th>course</th>
<th>ta</th>
</tr>
</thead>
<tbody>
<tr>
<td>PL</td>
<td>Ahuva</td>
</tr>
<tr>
<td>OS</td>
<td>Asma</td>
</tr>
<tr>
<td>AI</td>
<td>Avner</td>
</tr>
</tbody>
</table>

And for the consistent database $ST$:

<table>
<thead>
<tr>
<th>student</th>
<th>track</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ahuva</td>
<td>SWEng</td>
</tr>
<tr>
<td>Asma</td>
<td>DataEng</td>
</tr>
<tr>
<td>Asma</td>
<td>BioInf</td>
</tr>
<tr>
<td>Alon</td>
<td>BioInf</td>
</tr>
</tbody>
</table>

The repairs are consistent according to $CT[t_a] \subseteq ST[\text{student}]$ and $ST: student \rightarrow track$. 

$CT[t_a] \subseteq ST[\text{student}]$

$ST : student \rightarrow track$
### Repair Examples

**I:**

<table>
<thead>
<tr>
<th>course</th>
<th>ta</th>
</tr>
</thead>
<tbody>
<tr>
<td>PL</td>
<td>Ahuva</td>
</tr>
<tr>
<td>OS</td>
<td>Asma</td>
</tr>
<tr>
<td>AI</td>
<td>Avner</td>
</tr>
</tbody>
</table>

**ST**

<table>
<thead>
<tr>
<th>student</th>
<th>track</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ahuva</td>
<td>SWEng</td>
</tr>
<tr>
<td>Asma</td>
<td>DataEng</td>
</tr>
<tr>
<td>Asma</td>
<td>BioInf</td>
</tr>
<tr>
<td>Alon</td>
<td>BioInf</td>
</tr>
</tbody>
</table>

\[ \text{CT}[\text{ta}] \subseteq \text{ST}[\text{student}] \]

\[ \text{ST} : \text{student} \rightarrow \text{track} \]

\[ \Delta(I, J) = \{ \text{CT}(\text{AI}, \text{Avner}), \text{ST}(\text{Asma}, \text{BioInf}) \} \]
### Repair Examples

**I:**

<table>
<thead>
<tr>
<th>course</th>
<th>ta</th>
</tr>
</thead>
<tbody>
<tr>
<td>PL</td>
<td>Ahuva</td>
</tr>
<tr>
<td>OS</td>
<td>Asma</td>
</tr>
<tr>
<td>AI</td>
<td>Avner</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>student</th>
<th>track</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ahuva</td>
<td>SWEng</td>
</tr>
<tr>
<td>Asma</td>
<td>DataEng</td>
</tr>
<tr>
<td>Asma</td>
<td>BioInf</td>
</tr>
<tr>
<td>Alon</td>
<td>BioInf</td>
</tr>
</tbody>
</table>

\[CT[\text{ta}] \subseteq ST[\text{student}]\]

\[\text{ST} : \text{student} \rightarrow \text{track}\]

---

**J:**

<table>
<thead>
<tr>
<th>course</th>
<th>ta</th>
</tr>
</thead>
<tbody>
<tr>
<td>PL</td>
<td>Ahuva</td>
</tr>
<tr>
<td>OS</td>
<td>Asma</td>
</tr>
<tr>
<td>AI</td>
<td>Avner</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>student</th>
<th>track</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ahuva</td>
<td>SWEng</td>
</tr>
<tr>
<td>Asma</td>
<td>BioInf</td>
</tr>
<tr>
<td>Alon</td>
<td>BioInf</td>
</tr>
<tr>
<td>Avner</td>
<td>SWEng</td>
</tr>
</tbody>
</table>

\[\Delta(I,J) = \{\text{ST(Asma,DataEng)}, \text{ST(Avner,SWEng)}\}\]
Example of a non-Repair

\[ \Delta(I, J) = \{ CT(AI, Avner), ST(Asma, BioInf), ST(Asma, DataEng), ST(Asma, SWEng) \} \]
Subset Repairs

- Let $S = (\mathcal{R}, \Sigma)$ be a schema
- A constraint $\sigma \in \Sigma$ is **anti-monotone** if its satisfaction is preserved in subsets; formally, for every databases $J$ and $J'$ over $\mathcal{R}$ we have:

  $$J \models \sigma \text{ and } J' \subseteq J \Rightarrow J' \models \sigma$$

- We say that $\Sigma$ is **anti-monotone** if each of its members is anti-monotone
  - Hence, if $\Sigma$ is anti-monotone then its satisfaction is preserved in sub-instances
Examples of Anti-Monotone Constraints

- Functional dependencies
  - Why?
- Denial constraints $\forall x \neg (\varphi(x) \land \psi(x))$
  - Why?
- What about referential constraints $R[A] \subseteq S[B]$?
**Proposition**

Let $S = (\mathcal{R}, \Sigma)$ be a schema such that $\Sigma$ is anti-monotone, and $I$ an inconsistent database over $S$. Then every repair of $I$ is a subinstance of $I$; that is, if $J \in \text{Repairs}_\Sigma(I)$ then $J \subseteq I$. 
Recall: an independent set in a graph is a set of nodes that does not contain any edge.

Let $S = (R, \Sigma)$ be a schema, such that $\Sigma$ consists of only functional dependencies, and let $I$ be an inconsistent database over $S$.

A repair can be viewed as a maximal independent set of a graph.

Which graph?

Nodes are the facts of $I$, edges $\{f, g\}$ whenever $f$ and $g$ violate an FD in $\Sigma$. 

General Anti-Monotone Constraints

- In the case of more general anti-monotone constraints (e.g., DCs), we use the concept of a *hypergraph* (where an edge is any set of items, not necessarily pairs)
  - It is called the *conflict hypergraph*
- What is an *independent set of a hypergraph*?
Exercise: Counting Repairs 1

- The following signature $\mathcal{R}$ has a single relation symbol:

  $\text{Act}(actor, email, movie, role)$

- Suppose that $\Sigma$ consists of the following FDs:

  $actor \rightarrow email$

- Suggest an algorithm for counting the repairs of an inconsistent database $I$ over
Exercise: Counting Repairs 2

- The following signature $R$ has a single relation symbol:

  \[ \text{Act}(\text{actor}, \text{email}, \text{movie}, \text{role}) \]

- Suppose that $\Sigma$ consists of the following FDs:

  \[ \text{actor} \rightarrow \text{email} \]
  \[ \text{actor} \text{ movie} \rightarrow \text{role} \]

- Suggest an algorithm for counting the repairs of an inconsistent database $I$ over $(R, \Sigma)$
Exercise: Counting Repairs 3

- The following problem is \#P-hard: given a bipartite graph, count the maximal matches.
- Consider the signature \( \mathcal{R} \) with the single relation symbol:
  
  \[ \text{CEO}(\text{company}, \text{person}) \]

- Suggest FDs so that you can translate the problem of counting maximal matches into the problem of counting repairs.
# Table of Contents

1. Introduction
2. Inconsistent Databases
3. Repairs
4. **Consistent Query Answering**
5. Complexity Aspects
Recalling Database Queries

- Let $\mathcal{S} = (R, \Sigma)$ be a schema.
- Recall that a query $Q$ over $\mathcal{S}$ is associated with a heading $(A_1, \ldots, A_k)$, which is a sequence of distinct attributes.
- $Q$ maps every database $I \in \text{Inst}(\mathcal{S})$ into a relation $Q(I)$ over the heading of $Q$.
- A query with an empty heading is called Boolean, and we denote $Q(I)$ as either true or false.
**Definition (Consistent Answers)**

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

$$\text{Consistent}_\Sigma(Q, I) = \bigcap_{J \in \text{Repairs}_\Sigma(I)} Q(J)$$
CQA Examples

\[ I: \]

\[
\begin{array}{c|c}
\text{course} & \text{ta} \\
\hline
\text{PL} & \text{Ahuva} \\
\text{OS} & \text{Asma} \\
\text{AI} & \text{Avner} \\
\end{array}
\]

\[
\begin{array}{c|c}
\text{student} & \text{track} \\
\hline
\text{Ahuva} & \text{SWEng} \\
\text{Asma} & \text{DataEng} \\
\text{Asma} & \text{BioInf} \\
\text{Alon} & \text{BioInf} \\
\end{array}
\]

\[ \text{CT[ta]} \subseteq \text{ST[student]} \]

\[ \text{ST : student} \rightarrow \text{track} \]

- **Courses and tracks of their TAs**
  - (PL, SWEng)
- **All courses**
  - PL, OS
### More Interesting Example

$I:$

<table>
<thead>
<tr>
<th>lecturer</th>
<th>course</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avia</td>
<td>AI</td>
</tr>
<tr>
<td>Avia</td>
<td>DB</td>
</tr>
<tr>
<td>Aharon</td>
<td>DB</td>
</tr>
</tbody>
</table>

$\text{LC: lecturer } \rightarrow \text{ course}$

$TC:$

<table>
<thead>
<tr>
<th>ta</th>
<th>course</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ahuva</td>
<td>AI</td>
</tr>
<tr>
<td>Ahuva</td>
<td>DB</td>
</tr>
<tr>
<td>Asma</td>
<td>DB</td>
</tr>
</tbody>
</table>

$\text{TC: ta } \rightarrow \text{ course}$

$\text{TC: course } \rightarrow \text{ ta}$

**Which lecturers have a TA?**
In the case of a Boolean query $Q$, CQA boils down to “is $Q$ true in every repair?”

As usual, Boolean CQs are useful for complexity analysis.
**Boolean CQA Example**

\[ I: \]

<table>
<thead>
<tr>
<th></th>
<th>CT</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>course</td>
<td>ta</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PL</td>
<td>Ahuva</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OS</td>
<td>Asma</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AI</td>
<td>Avner</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>ST</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>student</td>
<td>track</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ahuva</td>
<td>SWEng</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Asma</td>
<td>DataEng</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Asma</td>
<td>BioInf</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alon</td>
<td>BioInf</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ CT[ta] \subseteq ST[\text{student}] \]

\[ ST: \text{student} \rightarrow \text{track} \]

- **Is there any TA from BioInf?**
- **Can we find at least two tracks with TAs?**
Table of Contents

1. Introduction
2. Inconsistent Databases
3. Repairs
4. Consistent Query Answering
5. Complexity Aspects
The literature of inconsistent databases studies several computational problems:

- Repair Checking
- Consistent Query Answering (CQA)
- Construction of a “good” repair (cleaning)
- Repair counting
- Repair enumeration
**Problem Def. (Repair Checking)**

Let $S = (R, \Sigma)$ be a schema. *Repair checking* is the problem of deciding, given an inconsistent database $I$ and a consistent database $J$, whether $J$ is a repair of $I$.

In notation: given $I \in \text{Inst}(R)$ and $J \in \text{Inst}(S)$, determine whether $J \in \text{Repairs}_\Sigma(I)$. 
Easy Exercise

- Let $S = (R, \Sigma)$ be a schema
- Suppose that both of the following hold:
  - $\Sigma$ is anti-monotone
  - $J \models \Sigma$ can be tested in polynomial time, given a database $J$ over $R$
- Prove that repair checking is solvable in polynomial time
Example of Intractable Repair Checking

**Theorem**

Let $S$ be the schema with the relation $R(A, B, C, D)$ and the constraints $R : A \rightarrow B$ and $R[C] \subseteq R[D]$. Then repair checking is coNP-complete over $S$.

- Proof by reduction from (the complement of) CNF-SAT
- Input of CNF-SAT is a formula $\varphi = c_0 \land \cdots \land c_{m-1}$
  - Each $c_i$ is a disjunction $l^1_i \lor \cdots \lor l^k_i$ of literals
  - A *literal* is either a variable $x$ (positive) or a negated variable $\neg x$ (negative)
  - Example: $\varphi = (x \lor y \lor z) \land (x \lor \neg y \lor w) \land (\neg x \lor \neg z \lor w)$
- Goal: is there any truth assignment that satisfies $\varphi$?
Proof (from [CM05])

- $R(A, B, C, D)$ with $R : A \rightarrow B$ and $R[C] \subseteq R[D]$  
- Given $\varphi = c_0 \land \cdots \land c_{m-1}$, we construct $I$ and $J$  
- $I$ contains:
  - $R(x, 1, (i + 1) \mod m, i)$ for every clause $c_i$ with the positive literal $x$  
  - $R(x, 0, (i + 1) \mod m, i)$ for every clause $c_i$ with the negative literal $\neg x$  
- $J$ is empty
Proof (from [CM05])

\[ R(A, B, C, D) \text{ with } R : A \rightarrow B \text{ and } R[C] \subseteq R[D] \]

Example: \( \varphi = (x \lor y \lor z) \land (x \lor \neg y \lor w) \land (\neg x \lor \neg z \lor w) \)

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>y</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>z</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>x</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>y</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>w</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>x</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>z</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>w</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>
Proof (from [CM05])

\[ R(A, B, C, D) \] with \( R : A \rightarrow B \) and \( R[C] \subseteq R[D] \)

Example: \( \varphi = (x \lor y \lor z) \land (x \lor \neg y \lor w) \land (\neg x \lor \neg z \lor w) \)

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( x )</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>( y )</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>( z )</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>( x )</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>( y )</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>( w )</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>( x )</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>( z )</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>( w )</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>
Proof (from [CM05])

- Every consistent subset of $I$ is either empty or encodes a satisfying assignment to $\varphi$
- Recall: $J$ is not a repair if and only if there exists a repair $J'$ such that $J' \neq J$ and $J' \leq_I J$
- Here, $\Delta(I, J) = I$ since $J$ is empty
- For a repair $J'$ we have $J' \leq_I J$ if and only if $\Delta(I, J') \subseteq \Delta(I, J)$
  - That is, $\Delta(I, J') \subseteq I$
  - That is, $J' \subseteq I$ (i.e., $J'$ is a subset repair)
- Hence, a repair $J' \neq J$ with $J' \leq_I J$ must be a nonempty consistent subset of $I$
- Hence, if such $J'$ exists (i.e., $J$ is not a repair), then $\varphi$ is satisfiable
Proof (from [CM05])

- The other direction is easy: if $\varphi$ is satisfiable, then we can construct a subset repair $J' \neq J$
- (Left as an exercise)
**Problem Def. (CQA)**

Let $\mathcal{S} = (\mathcal{R}, \Sigma)$ be a schema, and let $Q$ over a query over $\mathcal{S}$. CQA is the problem of deciding, given an inconsistent database $I$ and a tuple $a$, whether $a$ is a consistent answer.

In notation, given $I$ and $a$, is $a \in \text{Consistent}_\Sigma(Q, I)$?
Repair Checking vs. CQA

- CQA is motivated; but what about repair checking?
- Repair checking is viewed as an indirect indication of complexity: If we wish to manage inconsistency, the setup should be such that we should at least be able to test whether one database is the repair of another.
- There is also a more formal connection:
  - Suppose that:
    - repairs are of size polynomial in the inconsistent database;
    - query evaluation is in polynomial time.
  - If repair checking is in polynomial time, then CQA is in coNP
    - Why?
Example of a Tractable CQ

<table>
<thead>
<tr>
<th>( LC(\text{lecturer, course}) )</th>
<th>( CT(\text{course, ta}) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{lecturer} \rightarrow \text{course} )</td>
<td>( \text{course} \rightarrow \text{ta} )</td>
</tr>
</tbody>
</table>

\[ \begin{array}{|c|c|} \hline \text{lecturer} & \text{course} \rule{0pt}{2.6ex} \\ \hline \text{Keren} & \text{PL} \rule{0pt}{2.6ex} \\ \text{Keren} & \text{DB} \\ \text{Eran} & \text{PL} \\ \text{Eran} & \text{AI} \rule{0pt}{2.6ex} \\ \hline \end{array} \] \hspace{1cm} \begin{array}{|c|c|} \hline \text{course} & \text{ta} \rule{0pt}{2.6ex} \\ \hline \text{PL} & \text{Ahuva} \rule{0pt}{2.6ex} \\ \text{PL} & \text{Asma} \\ \text{AI} & \text{Avner} \\ \text{OS} & \text{Ahuva} \rule{0pt}{2.6ex} \\ \hline \end{array} \]

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

\[ Q() := L(\text{x, y}), C(\text{y, z}) \]
Example of a Tractable CQ

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

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

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

- \(Q\) is consistently true if and only if there is a lecturer such that every one of her courses is in CT

- \(\exists x \left[ (\exists y [\text{LC}(x, y)]) \land \forall y [\text{LC}(x, y) \rightarrow \exists z [\text{CT}(y, z)]] \right]\)

- An FO query can be evaluated straightforwardly in polynomial time (recall: data complexity)
Example of an Intractable CQ

\[
\begin{align*}
\text{LC}(\text{lecturer, course}) & \quad | \quad \text{TC}(\text{ta, course}) \\
\text{lecturer} \rightarrow \text{course} & \quad | \quad \text{ta} \rightarrow \text{course}
\end{align*}
\]

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

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

We now show that answering \(Q\) is coNP-complete.
Proof of Hardness (from [CM05])

\[ \text{QLC}(\text{lecturer}, \text{course}) \mid \text{TC}(\text{ta}, \text{course}) \]

\[ \text{lecturer} \rightarrow \text{course} \mid \text{ta} \rightarrow \text{course} \]

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

- Proof by reduction from (the complement of) non-mixed CNF-SAT
- Input is a CNF \( \varphi = c_1 \land \cdots \land c_m \) where in each clause \( c_i \) either all literals are positive (positive clause) or all literals are negative (negative clause)
- Example:

\[ \varphi = (x \lor y) \land (w \lor z) \land (\neg x \lor \neg y \lor \neg w) \]
Reduction

- Given $\varphi$, we build $I$:
  - $LC(i, z)$ for each **positive** $c_i$ containing $z$
  - $TC(i, z)$ for each **negative** $c_i$ containing $\neg z$
Example

lecturer → course    ta → course

ϕ = (x ∨ y) ∧ (w ∨ z) ∧ (¬x ∨ ¬y ∨ ¬w) ∧ (¬x ∨ ¬z)

<table>
<thead>
<tr>
<th>LC</th>
<th>course</th>
<th>TC</th>
<th>course</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>ta</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>x</td>
<td>3</td>
<td>x</td>
</tr>
<tr>
<td>1</td>
<td>y</td>
<td>3</td>
<td>y</td>
</tr>
<tr>
<td>2</td>
<td>w</td>
<td>3</td>
<td>w</td>
</tr>
<tr>
<td>2</td>
<td>z</td>
<td>4</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>z</td>
</tr>
</tbody>
</table>
Example

$$\varphi = (x \lor y) \land (w \lor z) \land (\neg x \lor \neg y \lor \neg w) \land (\neg x \lor \neg z)$$

<table>
<thead>
<tr>
<th>LC</th>
<th>course</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>x</td>
</tr>
<tr>
<td>1</td>
<td>y</td>
</tr>
<tr>
<td>2</td>
<td>w</td>
</tr>
<tr>
<td>2</td>
<td>z</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TC</th>
<th>course</th>
</tr>
</thead>
<tbody>
<tr>
<td>ta</td>
<td>course</td>
</tr>
<tr>
<td>3</td>
<td>x</td>
</tr>
<tr>
<td>3</td>
<td>y</td>
</tr>
<tr>
<td>3</td>
<td>w</td>
</tr>
<tr>
<td>4</td>
<td>x</td>
</tr>
<tr>
<td>4</td>
<td>z</td>
</tr>
</tbody>
</table>
**Another Example of a Tractable CQ**

\[
\begin{align*}
\text{LC(lecturer, course)} & \quad \text{CT(course, ta)} \\
\text{lecturer} \rightarrow \text{course} & \quad \text{course} \rightarrow \text{ta}
\end{align*}
\]

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

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

Wijsen [Wij10] has proved:

- \(Q\) is **not** expressible in FO
- However, \(Q\) can be evaluated in polynomial time


End of lecture 7

Inconsistent Databases