Principles of Managing Uncertain Data

Lecture 7: Inconsistent Databases

Principles of Managing Uncertain Data

Lecture 7: Inconsistent Databases

Inconsistent Databases

Repairs

Consistent Query Answering

Complexity Aspects

Introduction

Inconsistent Databases

Repairs

Consistent Query Answering

Complexity Aspects

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

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

[ABC99]
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]
- 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

Recalling Schemas

- A schema \( S \) 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 \( \mathcal{R} \)

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: \( R[\mathcal{A}_1, \ldots, \mathcal{A}_m] \subseteq S[R_1, \ldots , R_m] \) where \( \mathcal{A}_1, \ldots, \mathcal{A}_m \) are distinct attributes of \( R \) and \( \mathcal{B}_1, \ldots, \mathcal{B}_m \) are distinct attributes of \( S \)
    - In the case of \( m = 1 \) it is called a referential constraint
Inconsistent Databases

Definition (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>Course</th>
<th>TA</th>
</tr>
</thead>
<tbody>
<tr>
<td>CT</td>
<td></td>
</tr>
<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>ST</td>
<td></td>
</tr>
<tr>
<td>Ahuva</td>
<td>SWEng</td>
</tr>
<tr>
<td>Asma</td>
<td>DataEng</td>
</tr>
<tr>
<td>Asma</td>
<td>BiolInf</td>
</tr>
<tr>
<td>Alon</td>
<td>BiolInf</td>
</tr>
</tbody>
</table>

Constraints:
- $CT[TA] \subseteq ST[Student]$
- This is a referential constraint
- $ST: student \rightarrow track$
- This is an FD and a key constraint

Example 1

$\Delta(I, J) = \{CT(AI, Avner), ST(Asma, BiolInf)\}$

Example 2

$\Delta(I, J) = \{ST(Asma, DataEng), ST(Avner, SWEng)\}$
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 $J_1 \succeq J_2$, if $\Delta(I, J_1) \subseteq \Delta(I, J_2)$.

$\succeq$ 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 $\text{Inst}(S)$ be the set of all the (consistent) databases over $S$. A repair of $I$ is a member of $\text{Inst}(S)$ that is minimal under $\succeq$.

We denote by $\text{Repairs}_S(I)$ the set of all the repairs of $I$.

---

**Example of a non-Repair**

$I$: 

<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>Al</td>
<td>Aver</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>Al</td>
<td>Aver</td>
</tr>
</tbody>
</table>

$$\Delta(I, J) = \{\text{ST}(\text{Asma,DataEng}), \text{ST}(\text{Aver,SWEng})\}$$

---

**Example of a non-Repair**

$I$: 

<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>Al</td>
<td>Aver</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>Al</td>
<td>Aver</td>
</tr>
</tbody>
</table>

$$\Delta(I, J) = \{\text{ST}(\text{Aver,SWEng})\}$$

---

**Example of a non-Repair**

$I$: 

<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>Al</td>
<td>Aver</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>Al</td>
<td>Aver</td>
</tr>
</tbody>
</table>

$$\Delta(I, J) = \{\text{ST}(\text{Aver,SWEng})\}$$
Introduction

Inconsistent Databases

Repairs

Consistent Query Answering

Complexity Aspects

References

Subset Repairs

Examples of Anti-Monotone Constraints

Proposition

Let \( S = (R, \Sigma) \) be a schema such that \( \Sigma \) is anti-monotone, and \( I \) an inconsistent database over \( S \). Then every repair of \( I \) is a sub-instance of \( I \); that is, if \( J \in \text{Repairs}_{\Sigma}(I) \) then \( J \subseteq I \).
Exercise: Counting Repairs 2

- The following signature $\mathcal{R}$ has a single relation symbol:
  $\text{Act}(\text{actor, email, movie, role})$
- Suppose that $\Sigma$ consists of the following FDs:
  \[ \text{actor} \rightarrow \text{email} \]
  \[ \text{actor movie} \rightarrow \text{role} \]
- Suggest an algorithm for counting the repairs of an inconsistent database $I$ over $(\mathcal{R}, \Sigma)$

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:
  $\text{CEO}(\text{company, person})$
- Suggest FDs so that you can translate the problem of counting maximal matches into the problem of counting repairs

Dichotomy of Repair Counting

Theorem [LK17]

Let $S = (\mathcal{R}, \Sigma)$ be a fixed schema such that $\Sigma$ consists of only FDs.
- If the left sides of the FDs form a chain w.r.t. $\subseteq$ (up to equiv.), then the repairs can be counted in polynomial time.
- Otherwise, computing $|\text{Repairs}_S(I)|$ is #P-complete.

Recalling Database Queries

- Let $S = (R, \Sigma)$ be a schema
- Recall that a query $Q$ over $S$ is associated with a heading $(A_1, \ldots, A_k)$, which is a sequence of distinct attributes
- $Q$ maps every database $I \in \text{Inst}(S)$ 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

Consistent Answers

**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 $a \in Q(I)$ for every repair $J$. We denote by $\text{Consistent}_S(Q, I)$ the set of all consistent answers. Hence, we have:

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

I: CT course ta ST student track
   PL Ahuva Asma SWEng
   AI Avner

- Courses and tracks of their TAs
  - (PL, SWEng)
- All courses
  - PL, OS

ST: student → track

CONCEPTS

More Interesting Example

I: LC lecturer course TC ta course
   Avia AI Ahuva AI
   Avia DB Ahuva DB
   Aharon DB Asma DB

ST: student → track

Which lecturers have a TA?

Boolean CQA Example

I: CT course ta ST student track
   PL Ahuva Asma SWEng
   AI Avner

ST: student → track

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

Computational Problems

- 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 Definition (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 \( J \in \text{Repairs}_{S}(I) \).

**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 \( l_1 \lor \cdots \lor l_k \) 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

**Example**

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

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>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>( z )</td>
<td>0</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>

**Example**

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

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>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>( z )</td>
<td>0</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>
Example of a Tractable CQ

**Consistent Query Answering**

**Problem Def. (CQA)**

Let $S = (R, \Sigma)$ be a schema, and let $Q$ over a query over $S$. 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 $a \in \text{Consistent}_S(Q, I)$?

**Example of a Tractable CQ**

<table>
<thead>
<tr>
<th>$L$</th>
<th>$C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>lecturer, course</td>
<td>course $\rightarrow$ ta</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>lecturer</th>
<th>course</th>
<th>ta</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keren</td>
<td>PL</td>
<td>Ahuva</td>
</tr>
<tr>
<td>Eran</td>
<td>PL, AI</td>
<td>Avner, Ahuva</td>
</tr>
</tbody>
</table>

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

$Q() : L \equiv L(x, y), CT(y, z)$

- $Q$ is consistently true if and only if there is a lecturer such that every one of her courses is in $CT$.
- An FO query can be evaluated straightforwardly in polynomial time (recall: data complexity).
Example of an Intractable CQ

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

Query: Does any course have both a lecturer and a TA?
\[ Q() := \text{LC}(x, y), \text{TC}(x', y) \]

- We now show that answering \( Q \) is coNP-complete.

Proof of Hardness (from [CM05])

\[ \text{LC}(\text{lecturer}, \text{course}) \quad \text{TC}(\text{ta}, \text{course}) \]
\[ \text{lecturer} \rightarrow \text{course} \quad \text{ta} \rightarrow \text{course} \]
\[ Q() := \text{LC}(x, y), \text{TC}(x', y) \]

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

\[ \begin{array}{c|c|c}
& \text{lecturer} & \text{course} \\
\hline
1 & x & 3 \\
1 & y & 3 \\
2 & w & 3 \\
2 & z & 4 \\
\end{array} \quad \begin{array}{c|c}
& \text{ta} & \text{course} \\
\hline
3 & x & 4 \\
3 & y & 4 \\
3 & w & 4 \\
\end{array} \]

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

Example

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

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

\[ \begin{array}{c|c|c}
& \text{lecturer} & \text{course} \\
\hline
1 & x & 3 \\
1 & y & 3 \\
2 & w & 3 \\
2 & z & 4 \\
\end{array} \quad \begin{array}{c|c|c}
& \text{ta} & \text{course} \\
\hline
3 & x & 4 \\
3 & y & 4 \\
3 & w & 4 \\
\end{array} \]

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

Another Example of a Tractable CQ

\[ \text{LC}(\text{lecturer}, \text{course}) \quad \text{CT}(\text{course}, \text{ta}) \]
\[ \text{lecturer} \rightarrow \text{course} \quad \text{course} \rightarrow \text{ta} \]

Query: Does any course have the same lecturer and TA?
\[ Q() := \text{LC}(x, y), \text{CT}(y, x) \]

- We will see the proof of Kolaitis and Pema [KP12].

Wijsen [Wij10] has proved:
- \( Q \) is not expressible in FO
- However, \( Q \) can be evaluated in polynomial time
• 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 $CT(a',b')$
    • every two conflicting facts $CT(c,d)$ and $CT(c',d')$
    • every two joinable facts $LC(a,b)$ and $CT(b,a)$

**Lemma [KP12]**

Consider the CQ $Q(\cdot) : LC(x,y), CT(y,x)$ 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 $n$

**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
Completing the Proof

- Lemma: \( G_{Q,I} \) is claw free.
- Corollary: for \( Q : y \leftarrow \text{LC}(x,y), \text{CT}(y,x) \) the consistency problem can be solved in polynomial time

References


End of lecture 7

Inconsistent Databases