Coping with Inconsistent Databases
Semantics, Algorithms, and Complexity

Phokion G. Kolaitis

University of California Santa Cruz
and
IBM Research - Almaden
Logic and Databases

- In 1969, Edgar (Tedd) F. Codd introduced the relational data model.
  - Since that time, there has been a continuous and extensive interaction between logic and databases.
  - In 2007, C.J. Date wrote that logic and databases are “inextricably intertwined”.

- Two main uses of logic in databases:
  - Logic is used as a database query language to express questions asked against databases.
  - Logic is used as a specification language to express integrity constraints in databases.
The Relational Data Model

- Relational Database
  - Collection \((R_1, \ldots, R_m)\) of finite relations (tables).
  - Relational structure \(A = (A, R_1, \ldots, R_m)\).
    In relational databases, the universe is not made explicit. Typically, one works with the active domain of the database.

- Relational Query Languages
  - Relational Algebra: Operations \(\pi, \sigma, \times, \cup, \setminus\)
  - Relational Calculus: (Safe) First-Order Logic
  - SQL: The standard commercial database query language based on relational algebra and relational calculus.
Conjunctive Queries

Definition

A *conjunctive query* is a query specified by a first-order formula of the form

$$\exists y_1 \cdots \exists y_m \varphi(x_1, \ldots, x_n, y_1, \ldots, y_m),$$

where $\varphi(x_1, \ldots, x_n, y_1, \ldots, y_m)$ is a conjunction of atoms.

Example

- **PATH-OF-LENGTH-3**($x_1, x_2$):
  $$\exists y_1 \exists y_2 (E(x_1, y_1) \land E(y_1, y_2) \land E(y_2, x_2))$$

- **TAUGHT-BY**($x_1, x_2$):
  $$\exists y (ENROLLS(x_1, y) \land TEACHES(x_2, y)).$$
Conjunctive Queries

Fact

▶ Conjunctive queries are among the most frequently asked queries against databases.
▶ SQL provides direct support for expressing conjunctive queries via the SELECT ... FROM ... WHERE ... construct.

Example

▶ ENROLLS(student, course), TEACHES(professor, course)
▶ SQL expression for TAUGHT-BY:
SELECT ENROLLS.student, TEACHES.professor
FROM ENROLLS, TEACHES
WHERE ENROLLS.course = TEACHES.course
Boolean Conjunctive Queries

Definition
A Boolean conjunctive query is a conjunctive query with no free variables, i.e., it is of the form
\[ \exists y_1 \cdots \exists y_m \varphi(y_1, \ldots, y_m), \]
where \( \varphi(y_1, \ldots, y_m) \) is a conjunction of atoms.

Example
- \( \exists x, y, z (E(x, y) \land E(y, z) \land E(z, x)) \) ("there is a triangle")
- \( \exists x, y (R(x, y) \land T(y, x)) \).

Definition (CONJUNCTIVE QUERY EVALUATION - CQE)
Given a database \( D \) and a Boolean conjunctive query \( q \), does \( D \models q \)? (i.e., is \( q \) true on \( D \)?)
CQE and SAT

Fact  CQE is a generalization of SAT
CQE and SAT

Fact  CQE is a generalization of SAT

Example  The following statements are equivalent:

1. $(P \lor Q \lor T) \land (\neg P \lor Q \lor T) \land (\neg P \lor \neg Q \lor T)$ is satisfiable.
2. $D \models \exists x, y, z (R_0(x, y, z) \land R_1(x, y, z) \land R_2(x, y, z))$, where $D = (R_0, R_1, R_2)$ and $R_0 = \{(0, 1)\}^3 \setminus \{(0, 0, 0)\}$, $R_1 = \{(0, 1)\}^3 \setminus \{(1, 0, 0)\}$, $R_2 = \{(0, 1)\}^3 \setminus \{(1, 1, 0)\}$. 
CQE and SAT

Fact CQE is a generalization of SAT

Example The following statements are equivalent:

1. \((P \lor Q \lor T) \land (\neg P \lor Q \lor T) \land (\neg P \lor \neg Q \lor T)\) is satisfiable.

2. \(D \models \exists x, y, z (R_0(x, y, z) \land R_1(x, y, z) \land R_2(x, y, z))\), where
   \(D = (R_0, R_1, R_2)\) and \(R_0 = \{(0, 1)\}^3 \setminus \{(0, 0, 0)\}\),
   \(R_1 = \{(0, 1)\}^3 \setminus \{(1, 0, 0)\}\), \(R_2 = \{(0, 1)\}^3 \setminus \{(1, 1, 0)\}\).

Fact There is a difference between CQE and \(k\)-SAT, \(k \geq 2\).

- **Data Complexity:** In CQE, the query is typically fixed, but
  the database varies.
  The **Data Complexity** of CQE is in \(L\).

- **Expression Complexity:** In \(k\)-SAT (viewed as a CQE problem), the query varies, but the database is fixed.
  The **Expression Complexity** of CQE is \(NP\)-complete.
Integrity Constraints in Relational Databases

Extensive study of various types of integrity constraints in relational databases during the 1970s and early 1980s:

- Key constraints and functional dependencies
- Inclusion dependencies, join dependencies, multi-valued dependencies, ...

Eventually, it was realized that all these different types of dependencies can be specified in fragments of first-order logic.
Two Unifying Classes of Integrity Constraints

Definition

- **Equality-generating dependency (egd):**
  \[ \forall x (\phi(x) \rightarrow x_i = x_j), \]
  where \( \phi(x) \) is a conjunction of atoms.

  **Special Cases:** Key constraints, functional dependencies.

- **Tuple-generating dependency (tgd):**
  \[ \forall x (\phi(x) \rightarrow \exists y \psi(x, y)), \]
  where \( \phi(x) \) is a conjunction of atoms with vars. in \( x \), and \( \psi(x, y) \) is a conjunction of atoms with vars. in \( x \) and \( y \).

  **Special Cases:** LAV constraints, GAV constraints.
Equality-Generating Dependencies

Definition

- **Functional Dependency** $R : X \rightarrow Y$
  If two tuples in $R$ agree on $X$, then they agree on $Y$.

- **Key Constraint** $R : X \rightarrow Y$, where $Y$ is the set of attributes of $R$ that are not in $X$.

Example $R(A, B, C, D)$

- **Functional Dependency** $R : A, B \rightarrow D$ as an egd:
  \[
  \forall a, b, c, c', d, d'(R(a, b, c, d) \land R(a, b, c', d') \rightarrow d = d')
  \]

- **Key Constraint** $R : A, B \rightarrow C, D$ as two egds:
  \[
  \forall a, b, c, c', d, d'(R(a, b, c, d) \land R(a, b, c', d') \rightarrow c = c')
  \]
  \[
  \forall a, b, c, c', d, d'(R(a, b, c, d) \land R(a, b, c', d') \rightarrow d = d')
  \]
Inconsistent Databases

- In designing databases, one specifies a schema $S$ and a set $\Sigma$ of integrity constraints on $S$.
- An inconsistent database is a database $I$ that does not satisfy $\Sigma$.
- Inconsistent databases arise in a variety of contexts and for different reasons:
  - For lack of support of particular integrity constraints.
  - In data integration of heterogeneous data obeying different integrity constraints.
  - In data warehousing and in Extract-Transform-Load (ETL) applications, where data has to be “cleaned” before it can be processed.
Coping with Inconsistent Databases

Two different approaches:

- **Data Cleaning**: Based on heuristics or specific domain knowledge, the inconsistent database is transformed to a consistent one by modifying (adding, deleting, updating) tuples in relations.
  - This is the main approach in industry (e.g., IBM InfoSphere Quality Stage, Microsoft DQS).
  - More engineering than science as quite often arbitrary choices have to be made.
Coping with Inconsistent Databases

Two different approaches:

- **Data Cleaning**: Based on heuristics or specific domain knowledge, the inconsistent database is transformed to a consistent one by modifying (adding, deleting, updating) tuples in relations.
  - This is the main approach in industry (e.g., IBM InfoSphere Quality Stage, Microsoft DQS).
  - More engineering than science as quite often arbitrary choices have to be made.

- **Database Repairs**: A framework for coping with inconsistent databases in a principled way and without “cleaning” dirty data first.
Database Repairs

Definition (Arenas, Bertossi, Chomicki – 1999)

Σ a set of integrity constraints and I an inconsistent database. A database J is a repair of I w.r.t. Σ if

- J is a consistent database (i.e., $J \models \Sigma$);
- J differs from I in a minimal way.
Definition (Arenas, Bertossi, Chomicki – 1999)

Σ a set of integrity constraints and I an inconsistent database. A database J is a repair of I w.r.t. Σ if
- J is a consistent database (i.e., J ⊨ Σ);
- J differs from I in a minimal way.

Fact

Several different types of repairs have been considered:
- Set-based repairs (subset, superset, ⊕-repairs).
- Cardinality-based repairs
- Attribute-based repairs
- Preferred repairs
Subset Repairs

Definition
Σ a set of integrity constraints and I an inconsistent database. J is a *subset-repair* of I w.r.t. Σ if
- J ⊂ I
- J |= Σ (i.e., J is consistent)
- there is no J’ such that J’ |= Σ and J ⊂ J’ ⊂ I.

Note
From now on, we will use the term *repair*, instead of the term subset repair.
Example

Key constraint
\[ \Sigma = \{ \forall x \forall y \forall ( R(x, y) \land R(x, z) \rightarrow y = z) \} \]

Database
\[ I = \{ R(a_1, b_1), R(a_1, b_2), R(a_2, b_1), R(a_2, b_2) \} \]

I has four (subset) repairs w.r.t. \( \Sigma \):

- \( J_1 = \{ R(a_1, b_1), R(a_2, b_1) \} \)
- \( J_2 = \{ R(a_1, b_1), R(a_2, b_2) \} \)
- \( J_3 = \{ R(a_1, b_2), R(a_2, b_1) \} \)
- \( J_4 = \{ R(a_1, b_2), R(a_2, b_2) \} \).

Exponentially many repairs, in general.
Consistent Query Answering (CQA)

Definition (Arenas, Bertossi, Chomicki)
Σ a set of integrity constraints, q a query, and I a database. The \textit{consistent answers of q on I w.r.t. }\Sigma\textit{ is the set}

\[ \text{CON}(q, I, \Sigma) = \bigcap \{ q(J) : J \text{ is a repair of } I \text{ w.r.t. } \Sigma \}. \]

Note:
- The motivation comes from the semantics of queries in the context of \textit{incomplete information} and \textit{possible worlds}.
- The consistent answers of q in I are the \textit{certain answers of q on I}, when the set of all possible worlds is the set of all repairs of I w.r.t. \Sigma.
Example (Revisited)

\[\Sigma = \{\forall x \forall y \forall z((R(x, y) \land R(x, z) \rightarrow y = z)\}\}
\]
\[I = \{R(a_1, b_1), R(a_1, b_2), R(a_2, b_1), R(a_2, b_2)\}\]

Recall that \(I\) has four repairs w.r.t. \(\Sigma\):

- \(J_1 = \{R(a_1, b_1), R(a_2, b_1)\}\), \(J_2 = \{R(a_1, b_1), R(a_2, b_2)\}\)
- \(J_3 = \{R(a_1, b_2), R(a_2, b_1)\}\), \(J_4 = \{R(a_1, b_2), R(a_2, b_2)\}\).

- If \(q(x)\) is the query \(\exists y R(x, y)\), then

\[\text{CON}(q, I, \Sigma) = \{a_1, a_2\}.
\]

- If \(q(x)\) is the query \(\exists z R(z, x)\), then

\[\text{CON}(q, I, \Sigma) = \emptyset.
\]
Main themes explored so far:

- **Complexity of CQA for conjunctive queries:**
  From polynomial-time computability to undecidability.

- **Repair Checking:** Given $I$ and $J$, is $J$ a repair of $I$ w.r.t. $\Sigma$?
  From polynomial-time computability to coNP-completeness.

- **Prototype CQA Systems** for selected classes of constraints and selected classes of queries (mainly, conjunctive queries).
Complexity of CQA: A “Simple” Case Study

Definition Assume that

- $\Sigma$ is a set of key constraints with one key per relation.
- $q$ is a Boolean conjunctive query (no free variables).

$CERTAINTY(q, \Sigma)$ is the following decision problem: Given a database $I$, is $CON(q, I, \Sigma)$ true? (i.e., is $q$ true on every repair of $I$?)

Fact

- Repair checking is in P (in fact, it is in L).
- $CERTAINTY(q, \Sigma)$ is in coNP.
Binary relations $R$ and $S$ having the first attribute as key, i.e.,

$$\Sigma = \{ R(u, v) \land R(u, w) \rightarrow v = w, \ S(u, v) \land S(u, w) \rightarrow v = w \}.$$ 

- Let $q_1$ be the Boolean query $\exists x, y, z (R(x, y) \land S(y, z)).$
- Let $q_2$ be the Boolean query $\exists x, y (R(x, y) \land S(y, x)).$
- Let $q_3$ be the Boolean query $\exists x, y, z (R(x, y) \land S(z, y)).$

**Question:**
What can we say about $\text{CERTAINTY}(q_i, \Sigma)$, where $i = 1, 2, 3$?
Complexity of CQA: An Illustration

Binary relations $R$ and $S$ having the first attribute as key, i.e.,

$$\Sigma = \{ R(u, v) \land R(u, w) \rightarrow v = w, \ S(u, v) \land S(u, w) \rightarrow v = w \}.$$  

- Let $q_1$ be the query $\exists x, y, z (R(x, y) \land S(y, z))$.
  \textsc{Certainty}(q_1, \Sigma) is in P; in fact, it is FO-rewritable as $\exists x, y, z (R(x, y) \land S(y, z) \land \forall y'(R(x, y') \rightarrow \exists z' S(y', z')))$.  

- Let $q_2$ be the query $\exists x, y (R(x, y) \land S(y, x))$.
  \textsc{Certainty}(q_2, \Sigma) is in P, but it is not FO-rewritable.  

- Let $q_3$ be the query $\exists x, y, z (R(x, y) \land S(z, y))$.
  \textsc{Certainty}(q_3, \Sigma) is coNP-complete.
Classifying the Complexity of CQA

**Question:** Can we classify the complexity of \( \text{CERTAINTY}(q, \Sigma) \)?
Classifying the Complexity of CQA

**Question:** Can we classify the complexity of \textsc{Certainty}(q, \Sigma)?

**Conjecture (Dichotomy Conjecture for \textsc{Certainty}(q, \Sigma))**

If \( \Sigma \) is a set of key constraints with one key per relation and \( q \) is a Boolean conjunctive query, then one of the following holds:

- \textsc{Certainty}(q, \Sigma) is in \( \mathbb{P} \).
- \textsc{Certainty}(q, \Sigma) is \( \text{coNP} \)-complete.

Moreover, the dichotomy is **effective**: we can decide in PTIME whether \textsc{Certainty}(q, \Sigma) is in \( \mathbb{P} \) or it is \( \text{coNP} \)-complete.
Ladner’s Theorem and Dichotomies in Complexity

Theorem (Ladner - 1975)
If \( P \neq NP \), then there is a decision problem \( Q \) such that
- \( Q \) is in \( NP \), but not in \( P \).
- \( Q \) is not \( NP \)-complete.

The Fine Structure of \( NP \)

<table>
<thead>
<tr>
<th>( NP )-complete</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>not ( NP )-complete, not in ( P )</td>
<td></td>
</tr>
<tr>
<td>( P )</td>
<td></td>
</tr>
</tbody>
</table>
Ladner’s Theorem and Dichotomies in Complexity

Theorem (Ladner - 1975)
If $P \neq NP$, then there is a decision problem $Q$ such that

- $Q$ is in $NP$, but not in $P$.
- $Q$ is not $NP$-complete.

The Fine Structure of $NP$

<table>
<thead>
<tr>
<th></th>
<th>NP-complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>not NP-complete</td>
<td>not in $P$</td>
</tr>
<tr>
<td>$P$</td>
<td></td>
</tr>
</tbody>
</table>

Dichotomy Conjecture for $CERTAINTY(q, \Sigma)$

<table>
<thead>
<tr>
<th></th>
<th>coNP-complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>not coNP-complete</td>
<td>not in $P$</td>
</tr>
<tr>
<td>$P$</td>
<td></td>
</tr>
</tbody>
</table>
Progress towards the Dichotomy for \textsc{Certainty}(q, \Sigma)

Theorem (Koutris and Wijsen - 2015)
If \(\Sigma\) is a set of key constraints with one key per relation and \(q\) is a Boolean self-join free conjunctive query, then one of the following holds:

\begin{itemize}
  \item \textsc{Certainty}(q, \Sigma) is in \text{P}.
  \item \textsc{Certainty}(q, \Sigma) is \text{coNP}-complete.
\end{itemize}

Moreover, this dichotomy is decidable in quadratic time.
Progress towards the Dichotomy for \textsc{Certainty}(q, \Sigma)

\textbf{Theorem (Koutris and Wijsen - 2015)}

If \( \Sigma \) is a set of key constraints with one key per relation and \( q \) is a Boolean self-join free conjunctive query, then one of the following holds:

\begin{itemize}
  \item \textsc{Certainty}(q, \Sigma) is in P.
  \item \textsc{Certainty}(q, \Sigma) is coNP-complete.
\end{itemize}

Moreover, this dichotomy is decidable in quadratic time.

\textbf{Key Notion:} The attack graph associated with \( \Sigma \) and \( q \).

\begin{itemize}
  \item The nodes of the attack graph are the atoms of \( q \).
  \item The edges of the attack graph are determined by the functional dependencies on the variables of an atom that are implied by the keys of the other atoms.
\end{itemize}
Progress towards the Dichotomy for \textsc{Certainty}(q, \Sigma)

Theorem (Koutris and Wijsen - 2015)
Let \(\Sigma\) be a set of key constraints with one key per relation and let \(q\) is a Boolean self-join free conjunctive query.

- If the attack graph is acyclic, then \(\textsc{Certainty}(q, \Sigma)\) is in \(\mathsf{P}\) and, in fact, it FO-rewritable;
  otherwise, \(\textsc{Certainty}(q, \Sigma)\) is \(\mathsf{L}\)-hard, hence it is not FO-rewritable.

- If the attack graph contains no strong cycle, then \(\textsc{Certainty}(q, \Sigma)\) is in \(\mathsf{P}\).

- If the attack graph contains a strong cycle, then \(\textsc{Certainty}(q, \Sigma)\) is \(\mathsf{coNP}\)-complete.

Moreover, these conditions can be checked in quadratic time.
Applying the Koutris-Wisjen Dichotomy Theorem

Theorem (K... and Pema - 2012)

Assume $\Sigma$ consists of a key for $R$ and a key for $S$, and let $q$ be a Boolean query with two atoms, one $R$-atom and one $S$-atom. If $\text{CERTAINTY}(q, \Sigma)$ is not FO-rewritable, then the following hold:

- If $\text{key}(R) \cup \text{key}(S) \subseteq \text{Var}(R) \cap \text{Var}(S)$, then $\text{CERTAINTY}(q, \Sigma)$ is in $P$.
- If $\text{key}(R) \cup \text{key}(S) \not\subseteq \text{Var}(R) \cap \text{Var}(S)$, then $\text{CERTAINTY}(q, \Sigma)$ is $\text{coNP}$-complete.

Examples:

- Let $q_2$ be the query $\exists x, y (R(x, y) \land S(y, x))$. $\text{CERTAINTY}(q_2, \Sigma)$ is in $P$, because $\text{key}(R) \cup \text{key}(S) = \{x, y\}$, $\text{Var}(R) \cap \text{Var}(S) = \{x, y\}$.

- Let $q_3$ be the query $\exists x, y, z (R(x, y) \land S(z, y))$. $\text{CERTAINTY}(q_3, \Sigma)$ is $\text{coNP}$-complete, because $\text{key}(R) \cup \text{key}(S) = \{x, z\}$, $\text{Var}(R) \cap \text{Var}(S) = \{y\}$. 
Applying the Koutris-Wisjen Dichotomy Theorem

Theorem (K... and Pema - 2012)

Assume \( \Sigma \) consists of a key for \( R \) and a key for \( S \), and let \( q \) be a Boolean query with two atoms, one \( R \)-atom and one \( S \)-atom. If \( \text{CERTAINTY}(q, \Sigma) \) is not FO-rewritable, then the following hold:

- If \( \text{key}(R) \cup \text{key}(S) \subseteq \text{Var}(R) \cap \text{Var}(S) \), then \( \text{CERTAINTY}(q, \Sigma) \) is in \( P \).
- If \( \text{key}(R) \cup \text{key}(S) \not\subseteq \text{Var}(R) \cap \text{Var}(S) \), then \( \text{CERTAINTY}(q, \Sigma) \) is coNP-complete.

Examples:

- Let \( q_2 \) be the query \( \exists x, y (R(x, y) \land S(y, x)) \).
  \( \text{CERTAINTY}(q_2, \Sigma) \) is in \( P \), because \( \text{key}(R) \cup \text{key}(S) = \{x, y\} \), \( \text{Var}(R) \cap \text{Var}(S) = \{x, y\} \).

- Let \( q_3 \) be the query \( \exists x, y, z (R(x, y) \land S(z, y)) \).
  \( \text{CERTAINTY}(q_3, \Sigma) \) is coNP-complete, because \( \text{key}(R) \cup \text{key}(S) = \{x, z\} \), \( \text{Var}(R) \cap \text{Var}(S) = \{y\} \).
Open Problems

▸ Prove the Dichotomy Conjecture for \textsc{Certainty}(q, \Sigma), where \Sigma is a set of keys, one for each relation, and q is an arbitrary Boolean conjunctive query.

▸ Prove a Dichotomy Theorem for \textsc{Certainty}(q, \Sigma), where \Sigma is a set of functional dependencies and q is a union of Boolean conjunctive queries.
Beyond Keys and Functional Dependencies

The Broader Classification Challenge:

Classify the complexity of CERTAINTY\((q, \Sigma)\), where \(q\) is a FO-query and \(\Sigma\) is a “well-behaved” set of egds and tgds.
Beyond Keys and Functional Dependencies

The Broader Classification Challenge:
Classify the complexity of $\text{CERTAINTY}(q, \Sigma)$, where $q$ is a FO-query and $\Sigma$ is a “well-behaved” set of egds and tgds.

Fontaine - 2015:
Discovered an $a\ priori$ unexpected connection between Consistent Query Answering and Constraint Satisfaction.
Beyond Keys and Functional Dependencies

The Broader Classification Challenge:
Classify the complexity of \( \text{CERTAINTY}(q, \Sigma) \), where \( q \) is a FO-query and \( \Sigma \) is a “well-behaved” set of egds and tgds.

Fontaine - 2015:
Discovered an \textit{a priori} unexpected connection between Consistent Query Answering and Constraint Satisfaction.

Theorem (Fontaine - 2015)
If the dichotomy theorem holds for \( \text{CERTAINTY}(q, \Sigma) \), where \( \Sigma \) is a finite set of Horn constraints and \( q \) is a union of Boolean conjunctive queries, then the dichotomy theorem holds for the family \( \text{CSP}(B) \) of constraint satisfaction problems, where \( B \) is a relational structure.
Note

- CQA has been criticized as being too conservative: too many repairs may imply too few answers.
- CQA does not differentiate between repairs: all repairs are treated as equals.
Pragmatics of Consistent Query Answering

Note

- CQA has been criticized as being too **conservative**: too many repairs may imply too few answers.
- CQA does **not** differentiate between repairs: all repairs are treated as equals.

Staworko, Chomicki, and Marcinkowski - 2012

Introduced **prioritized repairing** that incorporates **preferences** between facts: if facts $f$ and $g$ **conflict**, we may prefer to resolve the **conflict** by deleting $g$ (and not $f$).

- $f$ may come from a more **reliable** source.
- $f$ may be more **current**.
Definition: Let $\Sigma$ be a set of functional dependencies (FDs). An inconsistent prioritizing database is a pair $(I, \succ)$, where

- $I$ is an inconsistent database w.r.t. $\Sigma$.
- $\succ$ is an acyclic binary relation on the facts of $I$ such that if $f \succ g$, then $f$ and $g$ violate one of the FDs in $\Sigma$.

Intuition:

- $f \succ g$ should be interpreted as “between the conflicting facts $f$ and $g$, we prefer to keep $f$ rather than $g$”.
- A preference relation between conflicting facts induces a preference relation between repairs.
- Thus, we can focus on “optimally preferred” repairs.
Globally Optimal Repairs

Definition (Staworko, Chomicki, Marcinkowski - 2012)

Σ set of FDs, \((I, \succ)\) an inconsistent prioritizing database.

- If \(J, K\) are two different consistent sub-databases of \(I\), then \(J\) is a global improvement of \(K\) if for every fact \(g \in K \setminus J\), there is a fact \(f \in J \setminus K\) such that \(f \succ g\).

<table>
<thead>
<tr>
<th>(J \setminus K)</th>
<th>(f)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(J \cap K)</td>
<td></td>
</tr>
<tr>
<td>(K \setminus J)</td>
<td>(g)</td>
</tr>
</tbody>
</table>

- \(J\) is a globally optimal repair of \(I\) (in short, a \(g\)-repair of \(I\)) if \(J\) is consistent and has no global improvement.

Note: Every \(g\)-repair of \((I, \succ)\) is a (subset) repair of \(I\).
course, term $\rightarrow$ instructor and instructor, term $\rightarrow$ course

<table>
<thead>
<tr>
<th></th>
<th>course</th>
<th>term</th>
<th>instructor</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_1$</td>
<td>DB</td>
<td>Fall</td>
<td>Anna</td>
</tr>
<tr>
<td>$f_2$</td>
<td>DB</td>
<td>Fall</td>
<td>Elsa</td>
</tr>
<tr>
<td>$f_3$</td>
<td>PL</td>
<td>Fall</td>
<td>Elsa</td>
</tr>
<tr>
<td>$f_4$</td>
<td>PL</td>
<td>Fall</td>
<td>Anna</td>
</tr>
<tr>
<td>$f_5$</td>
<td>PL</td>
<td>Spring</td>
<td>John</td>
</tr>
<tr>
<td>$f_6$</td>
<td>DB</td>
<td>Spring</td>
<td>John</td>
</tr>
<tr>
<td>$f_7$</td>
<td>PL</td>
<td>Spring</td>
<td>George</td>
</tr>
</tbody>
</table>

Preferences

- $f_2 \succ f_1$
- $f_4 \succ f_3$
- $f_5 \succ f_6$
- $f_5 \succ f_7$

$I$

<table>
<thead>
<tr>
<th></th>
<th>course</th>
<th>term</th>
<th>instructor</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_1$</td>
<td>DB</td>
<td>Fall</td>
<td>Anna</td>
</tr>
<tr>
<td>$f_3$</td>
<td>PL</td>
<td>Fall</td>
<td>Elsa</td>
</tr>
<tr>
<td>$f_4$</td>
<td>PL</td>
<td>Fall</td>
<td>Anna</td>
</tr>
<tr>
<td>$f_5$</td>
<td>PL</td>
<td>Spring</td>
<td>John</td>
</tr>
</tbody>
</table>

$K$ is a repair of $I$

<table>
<thead>
<tr>
<th></th>
<th>course</th>
<th>term</th>
<th>instructor</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_3$</td>
<td>PL</td>
<td>Fall</td>
<td>Elsa</td>
</tr>
<tr>
<td>$f_5$</td>
<td>PL</td>
<td>Spring</td>
<td>John</td>
</tr>
</tbody>
</table>

$J$ is a $g$-repair of $(I, \succ)$
Repair Checking

$\Sigma$ a fixed set of functional dependencies (FDs).

- **REPAIR CHECKING**: Given $I$ and $J$, is $J$ a repair of $I$?
- Recall that **REPAIR CHECKING** in P (in fact, it is in L).

**Definition**

$g$-**REPAIR CHECKING**: Given $(I, \succ)$ and $J$, is $J$ a $g$-repair of $I$?

- It is easy to see that $g$-**REPAIR CHECKING** is in coNP.
Repair Checking

\( \Sigma \) a fixed set of functional dependencies (FDs).

- **REPAIR CHECKING**: Given \( I \) and \( J \), is \( J \) a repair of \( I \)?
- Recall that **REPAIR CHECKING** is in \( \mathbb{P} \) (in fact, it is in \( \mathbb{L} \)).

**Definition**

\( g\text{-REPAIR CHECKING} \): Given \( (I, \succ) \) and \( J \), is \( J \) a \( g \)-repair of \( I \)?

- It is easy to see that \( g\text{-REPAIR CHECKING} \) is in \( \text{coNP} \).

**Theorem** (Staworko, Chomicki, Marcinkowski - 2012)
There is a set \( \Sigma \) of four FDs on a relation of arity 8 such that \( g\text{-REPAIR CHECKING} \) is \( \text{coNP} \)-complete.

**Question:**
Can we classify the complexity of \( g\text{-REPAIR CHECKING} \)?
Dichotomy Theorem for $g$-Repair Checking

Theorem (Fagin, Kimelfeld, K . . . - 2015)
Let $\Sigma$ be a set of FDs on a collection of relations.

- If $\Sigma$ induces a single FD or two key constraints on each relation, then $g$-REPAIR CHECKING is solvable in P.
- Otherwise, $g$-REPAIR CHECKING is coNP-complete.

Moreover, this dichotomy is effective.

Note
This is a data complexity result: the constraints are held fixed, the input consists of $(I, \succ)$ and $J$. 
Illustrating the Dichotomy for $g$-Repair Checking

Courses

<table>
<thead>
<tr>
<th>course</th>
<th>term</th>
<th>instructor</th>
</tr>
</thead>
</table>

Functional Dependencies

<table>
<thead>
<tr>
<th>Functional Dependencies</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>course, term $\rightarrow$ instructor</td>
<td>$P$</td>
</tr>
<tr>
<td>instructor, course $\rightarrow$ term</td>
<td>($two$ $keys$)</td>
</tr>
<tr>
<td>instructor $\rightarrow$ course</td>
<td>$P$</td>
</tr>
<tr>
<td></td>
<td>($one$ $FD$)</td>
</tr>
<tr>
<td>course $\rightarrow$ instructor</td>
<td>coNP-complete</td>
</tr>
<tr>
<td>instructor $\rightarrow$ course</td>
<td>($two$ $non$-$key$ $FDs$)</td>
</tr>
</tbody>
</table>
Proof Strategy for the Intractability Side

Two main steps:

1. Proof of intractability for six basic sets of FDs.
   All six basic sets of FDs are for a ternary relation $R(A, B, C)$:

   $$
   \begin{array}{|c|c|}
   \hline
   A \rightarrow B, B \rightarrow A & A \rightarrow B, B \rightarrow C \\
   \hline
   A \rightarrow B, C \rightarrow B & AB \rightarrow C, C \rightarrow B \\
   \hline
   AB \rightarrow C, AC \rightarrow B, BC \rightarrow A & \rightarrow A, B \rightarrow C \\
   \hline
   \end{array}
   $$

2. Proof of intractability for an arbitrary set of FDs,
   Use case analysis and distinct reductions from one of the six basic sets of FDs.
Open Problems for Preferred Repairs

- Classify the complexity of $g$-CERTAINTY($q, \Sigma$), where $q$ is a Boolean conjunctive query and $\Sigma$ is a set of FDs.
  - Is there a Trichotomy Theorem for $g$-CERTAINTY($q, \Sigma$)?
    (P, coNP-complete, $\Pi_2^P$-complete)

- What if the preference relation $\succ$ is specified syntactically?
  - Is there a “useful” language for expressing preferences such that $g$-repair checking and $g$-CERTAINTY($q, \Sigma$) are of lower complexity?
The framework of repairs and consistent query answering provides a principled approach to coping with inconsistency in databases.

Extensive study of the complexity of repair checking and consistent query answering during the past fifteen years.

This research, however, has not penetrated the practice of data cleaning.

One of the reasons for this gap between theory and practice is that industrial-strength CQA-systems have yet to be developed.
From Theory to Practice: Prototype CQA Systems

- Hippo (Chomicki, Marcinkowski, Staworko - 2004)
- ConQuer (Fuxman - 2007)
- ConsEx (Caniupan, Bertossi - 2010)
- EQUIP (K ... , Pema, Tan - 2013)
## Prototype CQA Systems: Features and Methods

<table>
<thead>
<tr>
<th>System</th>
<th>Constraints</th>
<th>Queries</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hippo</td>
<td>Universal</td>
<td>Projection-free with $\cup$ and $\setminus$</td>
<td>Direct Algorithm</td>
</tr>
<tr>
<td>ConQuer</td>
<td>Keys</td>
<td>Class of CQs</td>
<td>FO-rewriting</td>
</tr>
<tr>
<td>ConsEx</td>
<td>Universal + INC with acyclicity</td>
<td>Datalog with $\neg$</td>
<td>Answer Set Programming</td>
</tr>
<tr>
<td>EQUIP</td>
<td>Keys</td>
<td>Arbitrary CQs</td>
<td>Reduction to ILP</td>
</tr>
</tbody>
</table>
Protype CQA System EQUIP

- EQUIP computes \( \text{CON}(q, \Sigma) \), where \( q \) is an arbitrary conjunctive query and \( \Sigma \) is a set of key constraints.

- Main Ingredients: Reduction to ILP + Database Techniques

- Extensive experimentation with 21 conjunctive queries for which \( \text{CON}(q, \Sigma) \) spans all three possibilities: FO-rewritable, in \( \mathbb{P} \) but not FO-rewritable, \( \text{coNP} \)-complete.

- Comparison of EQUIP with both ConQuer and ConsEx:
  - ConQuer does better than EQUIP on conjunctive queries for which \( \text{CON}(q, \Sigma) \) is FO-rewritable.
  - EQUIP significantly outperforms ConsEX on conjunctive queries for which \( \text{CON}(q, \Sigma) \) is in \( \mathbb{P} \) but not FO-rewritable or \( \text{CON}(q, \Sigma) \) is \( \text{coNP} \)-complete.
EQUIP vs. ConQuer

Query Q15

Evaluation time (in seconds)

Query Q16

Number of tuples per relation (in thousands)

Query Q17

Evaluation time (in seconds)

Query Q18

Number of tuples per relation (in thousands)
EQUIP vs. ConsEx

Query Q1
Evaluation time (in seconds)
ConsEx
EQUIP

Query Q8

Query Q2
Evaluation time (in seconds)
ConsEx
EQUIP

Query Q9

Number of tuples per relation (in thousands)
# Queries for Evaluating EQUIP

<table>
<thead>
<tr>
<th>Complexity of CQA:</th>
<th>coNP-complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>( Q_1() ): ( - )</td>
<td>( R_5(x, y, z) ), ( R_6(x', y, w) )</td>
</tr>
<tr>
<td>( Q_2(z) ): ( - )</td>
<td>( R_5(x, y, z) ), ( R_6(x', y, w) )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Complexity of CQA:</th>
<th>in P, not FO-rewritable</th>
</tr>
</thead>
<tbody>
<tr>
<td>( Q_8() ): ( - )</td>
<td>( R_3(x, y, z) ), ( R_4(y, x, w) )</td>
</tr>
<tr>
<td>( Q_9(z) ): ( - )</td>
<td>( R_3(x, y, z) ), ( R_4(y, x, w) )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Complexity of CQA:</th>
<th>FO-rewritable</th>
</tr>
</thead>
<tbody>
<tr>
<td>( Q_{15}(z) ): ( - )</td>
<td>( R_1(x, y, z) ), ( R_2(y, v, w) )</td>
</tr>
<tr>
<td>( Q_{16}(z, w) ): ( - )</td>
<td>( R_1(x, y, z) ), ( R_2(y, v, w) )</td>
</tr>
<tr>
<td>( Q_{17}(z) ): ( - )</td>
<td>( R_1(x, y, z) ), ( R_2(y, v) ), ( R_7(v, u, d) )</td>
</tr>
<tr>
<td>( Q_{18}(z, w) ): ( - )</td>
<td>( R_1(x, y, z) ), ( R_2(y, v) ), ( R_7(v, u, d) )</td>
</tr>
</tbody>
</table>
A Vision for a Comprehensive CQA System

- **Preprocessing:** Use the complexity classification of $\text{CONS}(q, \Sigma)$ to determine the evaluation strategy.
  - Given $q$ and $\Sigma$, determine if $\text{CONS}(q, \Sigma)$ is FO-rewritable or in $P$ but not FO-rewritable or $\text{coNP}$-complete.

- **Module A:** $\text{CONS}(q, \Sigma)$ is FO-rewritable
  - FO-rewriting algorithm + Database Engine

- **Module B:** $\text{CONS}(q, \Sigma)$ is in $P$ but not FO-rewritable
  - Direct Algorithm or Reduction to Linear Programming + LP Solver

- **Module C:** $\text{CONS}(q, \Sigma)$ is $\text{coNP}$-hard (or harder)
  - Reduction to ILP (or to SAT or to QBF) + Solvers
Synopsis and Outlook

- The framework of repairs and consistent query answering is a meeting point of databases, logic, and computational complexity.

- While much progress has been made towards delineating the computational complexity of repair checking and consistent query answering, many challenges - in the form of dichotomy theorems - remain.

- Much remains to be done towards building comprehensive CQA-systems for different types of repairs and different classes of constraints.

- Combining database engines with SAT solvers and QBF solvers may be a promising approach towards this goal.