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 $\mathbf{A} = (A, R_1, \ldots, R_m)$.

    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 π, σ, $\times$, $\cup$, $\setminus$
  - 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):
  $$\exists y_1 \exists y_2 (E(x_1, y_1) \wedge E(y_1, y_2) \wedge E(y_2, x_2))$$
- TAUGHT-BY(x_1, x_2):
  $$\exists y (\text{ENROLLS}(x_1, y) \wedge \text{TEACHES}(x_2, y)).$$

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

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

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. $(P \vee Q \vee T) \wedge (\neg P \vee Q \vee T) \wedge (\neg P \vee \neg Q \vee T)$ is satisfiable.
2. $D \models \exists x, y, z(R_0(x, y, z) \wedge R_1(x, y, z) \wedge R_2(x, y, z))$, 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. $(P \vee Q \vee T) \wedge (\neg P \vee Q \vee T) \wedge (\neg P \vee \neg Q \vee T)$ is satisfiable.
2. $D \models \exists x, y, z(R_0(x, y, z) \wedge R_1(x, y, z) \wedge R_2(x, y, z))$, 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):
$$\forall \mathbf{x}(\phi(\mathbf{x}) \rightarrow x_i = x_j),$$
where $\phi(\mathbf{x})$ is a conjunction of atoms.

  Special Cases: Key constraints, functional dependencies.

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

  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:
  $\forall a, b, c, c', d, d'(R(a, b, c, d) \wedge R(a, b, c', d') \rightarrow d = d')$
- Key Constraint $R : A, B \rightarrow C, D$ as two egds:
  $\forall a, b, c, c', d, d'(R(a, b, c, d) \wedge R(a, b, c', d') \rightarrow c = c')$
  $\forall a, b, c, c', d, d'(R(a, b, c, d) \wedge R(a, b, c', d') \rightarrow d = d')$

Inconsistent Databases

- In designing databases, one specifies a schema **S** and a set Σ of integrity constraints on **S**.
- An inconsistent database is a database *I* that does not satisfy Σ.
- 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.

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.

Fact

Several different types of repairs have been considered:

- Set-based repairs (subset, superset, $\oplus$-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

- $J \subset I$
- $J \models \Sigma$ (i.e., J is consistent)
- there is no J' such that $J' \models \Sigma$ and $J \subset J' \subset I$.

Note

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

Subset Repairs

Example

Key constraint

$\Sigma = \{\forall x \forall y \forall ((R(x, y) \wedge R(x, z) \rightarrow y = z)\}$

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. Σ:

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

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 *consistent answers of q on I w.r.t. Σ* is the set

$$\textsc{Con}(q, I, \Sigma) = \bigcap \{q(J) : J \text{ is a repair of } I \text{ w.r.t. } \Sigma\}.$$

Note:

- The motivation comes from the semantics of queries in the context of incomplete information and possible worlds.
- The consistent answers of q in I are the *certain answers of q on I*, when the set of all possible worlds is the set of all repairs of I w.r.t. Σ.

Consistent Query Answering (CQA)

Example (Revisited)

$\Sigma = \{\forall x \forall y \forall z((R(x,y) \wedge R(x,z) \rightarrow y = z)\}$
$I = \{R(a_1, b_1), R(a_1, b_2), R(a_2, b_1), R(a_2, b_2)\}$

Recall that I has four repairs w.r.t. Σ:

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

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

$$\textsc{Con}(q, I, \Sigma) = \{a_1, a_2\}.$$

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

$$\textsc{Con}(q, I, \Sigma) = \emptyset.$$

Overview of Research on Database Repairs

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. Σ?
  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

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

CERTAINTY(q, Σ) is the following decision problem:
Given a database I, is CON(q, I, Σ) 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, Σ) is in coNP.

Complexity of CQA: An Illustration

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

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

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

Question:
What can we say about CERTAINTY(q_i, Σ), where $i = 1, 2, 3$?

Complexity of CQA: An Illustration

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

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

- Let q_1 be the query $\exists x, y, z(R(x,y) \wedge S(y,z))$.
  CERTAINTY(q_1, Σ) is in P; in fact, it is FO-rewritable as $\exists x, y, z(R(x,y) \wedge S(y,z) \wedge \forall y'(R(x,y') \rightarrow \exists z' S(y',z')))$.
- Let q_2 be the query $\exists x, y(R(x,y) \wedge S(y,x))$.
  CERTAINTY(q_2, Σ) is in P, but it is not FO-rewritable.
- Let q_3 be the query $\exists x, y, z(R(x,y) \wedge S(z,y))$.
  CERTAINTY(q_3, Σ) is coNP-complete.

Classifying the Complexity of CQA

Question: Can we classify the complexity of CERTAINTY(q, Σ)?

Classifying the Complexity of CQA

Question: Can we classify the complexity of CERTAINTY(q, Σ)?

Conjecture (Dichotomy Conjecture for CERTAINTY(q, Σ))

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

- CERTAINTY(q, Σ) is in P.
- CERTAINTY(q, Σ) is coNP-complete.

Moreover, the dichotomy is effective: we can decide in PTIME whether CERTAINTY(q, Σ) is in P or it is coNP-complete.

Ladner's Theorem and Dichotomies in Complexity

Theorem (Ladner - 1975)

If $\mathrm{P} \neq \mathrm{NP}$, 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

NP-complete
not NP-complete, not in P
P

Ladner's Theorem and Dichotomies in Complexity

Theorem (Ladner - 1975)

If $\mathrm{P} \neq \mathrm{NP}$, 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

NP-complete
not NP-complete, not in P
P

Dichotomy Conjecture for CERTAINTY(q, Σ)

CERTAINTY(q, Σ) $\nearrow$	coNP-complete
	not coNP-complete, not in P
$\searrow$	P

Progress towards the Dichotomy for CERTAINTY(q, Σ)

Theorem (Koutris and Wijsen - 2015)

If Σ 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:

- CERTAINTY(q, Σ) is in P.
- CERTAINTY(q, Σ) is coNP-complete.

Moreover, this dichotomy is decidable in quadratic time.

Progress towards the Dichotomy for CERTAINTY(q, Σ)

Theorem (Koutris and Wijsen - 2015)

If Σ 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:

- CERTAINTY(q, Σ) is in P.
- CERTAINTY(q, Σ) is coNP-complete.

Moreover, this dichotomy is decidable in quadratic time.

Key Notion: The attack graph associated with Σ and q.

- The nodes of the attack graph are the atoms of q.
- 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.

Progress towards the Dichotomy for CERTAINTY(q, Σ)

Theorem (Koutris and Wijsen - 2015)

Let Σ 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
  CERTAINTY(q, Σ) is in P and, in fact, it FO-rewritable;
  otherwise,
  CERTAINTY(q, Σ) is L-hard, hence it is not FO-rewritable.
- If the attack graph contains no strong cycle, then
  CERTAINTY(q, Σ) is in P.
- If the attack graph contains a strong cycle, then
  CERTAINTY(q, Σ) is coNP-complete.

Moreover, these conditions can be checked in quadratic time.

Applying the Koutris-Wisjen Dichotomy Theorem

Theorem (K . . . and Pema - 2012)

Assume Σ 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 CERTAINTY(q, Σ) is not FO-rewritable, then the following hold:

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

Applying the Koutris-Wisjen Dichotomy Theorem

Theorem (K . . . and Pema - 2012)

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

Examples:

- Let q_2 be the query $\exists x, y(R(x, y) \wedge S(y, x))$. 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 $\exists x, y, z(R(x, y) \wedge S(z, y))$. CERTAINTY$(q_3, \Sigma)$ is coNP-complete, because $\text{key}(R) \cup \text{key}(S) = \{x, z\}$, $\text{Var}(R) \cap \text{Var}(S) = \{y\}$.

Beyond the Koutris-Wijsen Dichotomy Theorem

Open Problems

- Prove the Dichotomy Conjecture for CERTAINTY(q, Σ), where Σ is a set of keys, one for each relation, and q is an arbitrary Boolean conjunctive query.
- Prove a Dichotomy Theorem for CERTAINTY(q, Σ), where Σ 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, Σ), where q is a FO-query and Σ is a "well-behaved" set of egds and tgds.

Beyond Keys and Functional Dependencies

The Broader Classification Challenge:

Classify the complexity of CERTAINTY(q, Σ), where q is a FO-query and Σ 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 CERTAINTY(q, Σ), where q is a FO-query and Σ is a "well-behaved" set of egds and tgds.

Fontaine - 2015:

Discovered an *a priori* unexpected connection between Consistent Query Answering and Constraint Satisfaction.

Theorem (Fontaine - 2015)

If the dichotomy theorem holds for CERTAINTY(q, Σ), where Σ is a finite set of Horn constraints and q is a union of Boolean conjunctive queries, then the dichotomy theorem holds for the family CSP($\mathbf{B}$) of constraint satisfaction problems, where $\mathbf{B}$ is a relational structure.

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.

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.

Prioritizing Inconsistent Databases

Definition: Let Σ 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. Σ.
- $\succ$ 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 Σ.

Intuition:

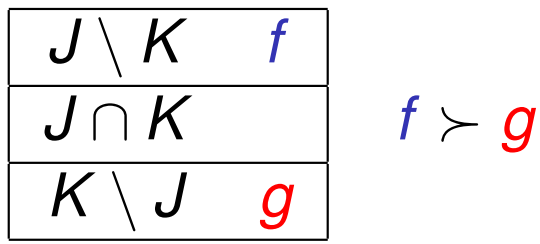
- $f \succ g$ should be interpreted as "between the conflicting facts f and g, we prefer to keep f rather than g".
- A preference relation between conflicting facts induces a preference relation between repairs.
- Thus, we can focus on "optimally preferred" repairs.

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 $f \succ g$.



- 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

I

	course	term	instructor
f_1	DB	Fall	Anna
f_2	DB	Fall	Elsa
f_3	PL	Fall	Elsa
f_4	PL	Fall	Anna
f_5	PL	Spring	John
f_6	DB	Spring	John
f_7	PL	Spring	George

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

K

	course	term	instructor
f_1	DB	Fall	Anna
f_3	PL	Fall	Elsa
f_5	PL	Spring	John

K is a repair of I

J

	course	term	instructor
f_2	DB	Fall	Elsa
f_4	PL	Fall	Anna
f_5	PL	Spring	John

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

Repair Checking

Σ 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

Σ 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.

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

Question:
Can we classify the complexity of g-REPAIR CHECKING?

Dichotomy Theorem for *g*-Repair Checking

Theorem (Fagin, Kimelfeld, K . . . - 2015)

Let Σ be a set of FDs on a collection of relations.

- If Σ 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

course	term	instructor

Functional Dependencies	Complexity
course, term $\rightarrow$ instructor instructor, course $\rightarrow$ term	P (two keys)
instructor $\rightarrow$ course	P (one FD)
course $\rightarrow$ instructor instructor $\rightarrow$ course	coNP-complete (two non-key FDs)

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 $R(A, B, C)$:

$A \to B$, $B \to A$	$A \to B$, $B \to C$
$A \to B$, $C \to B$	$AB \to C$, $C \to B$
$AB \to C$, $AC \to B$, $BC \to A$	$\to A$, $B \to C$

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, Σ), where q is a Boolean conjunctive query and Σ is a set of FDs.
  - Is there a Trichotomy Theorem for g-CERTAINTY(q, Σ)? (P, coNP-complete, Π_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, Σ) are of lower complexity?

Theory and Practice

- 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)

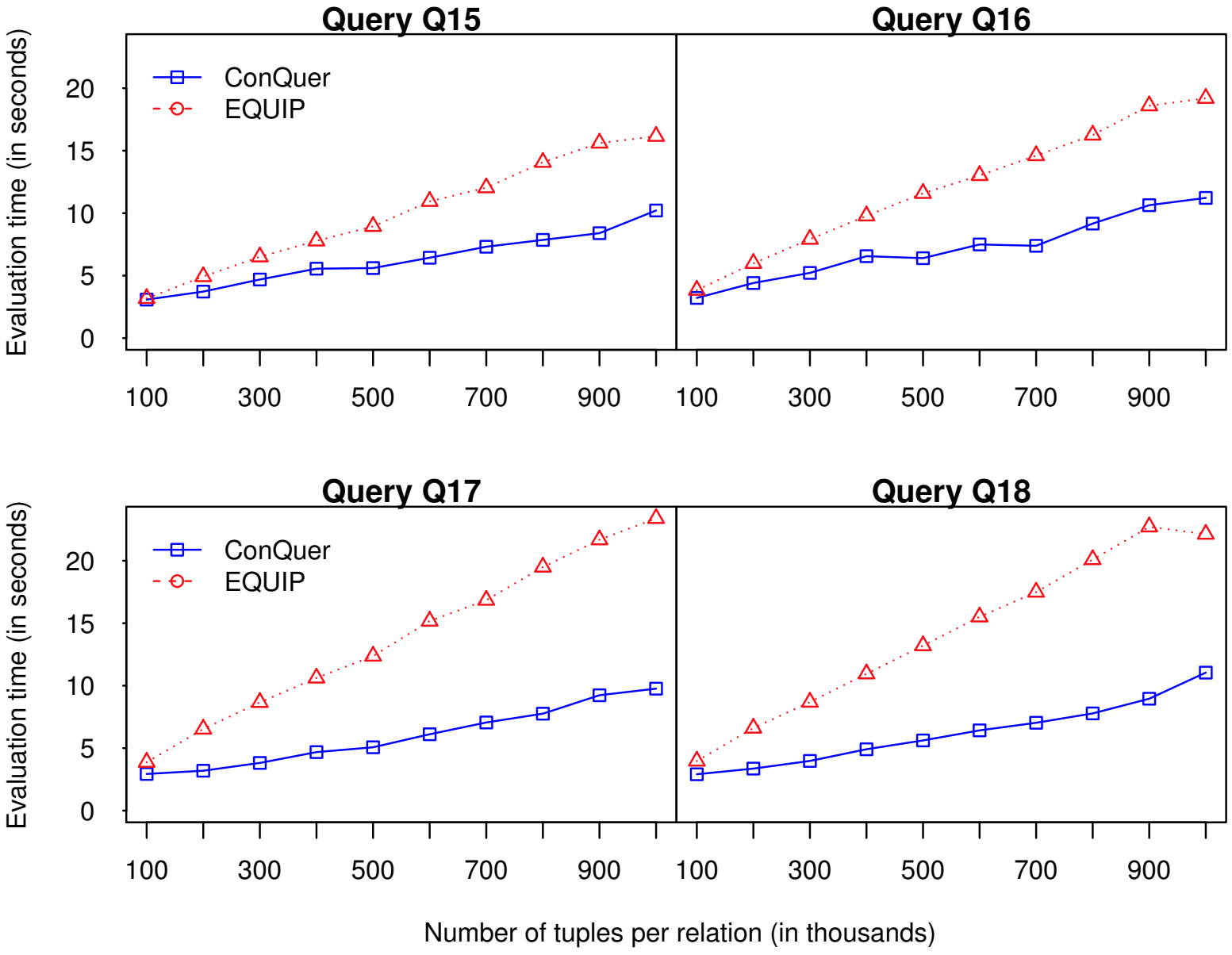
Prototype CQA Systems: Features and Methods

System	Constraints	Queries	Method
Hippo	Universal	Projection-free with $\cup$ and $\setminus$	Direct Algorithm
ConQuer	Keys	Class of CQs	FO-rewriting
ConsEx	Universal + INC with acyclicity	Datalog with $\neg$	Answer Set Programming
EQUIP	Keys	Arbitrary CQs	Reduction to ILP

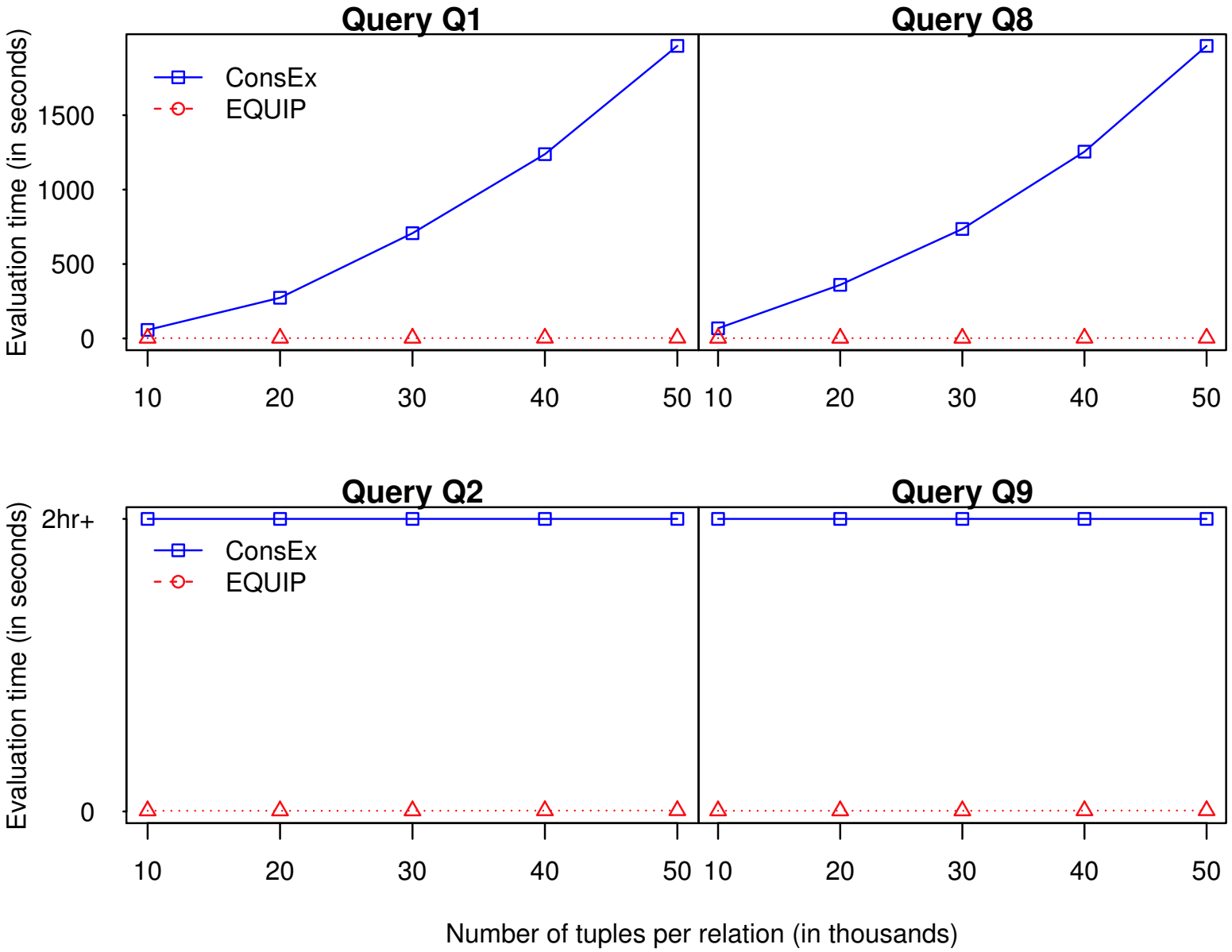
Protype CQA System EQUIP

- EQUIP computes $\textsc{Con}(q, \Sigma)$, where q is an arbitrary conjunctive query and Σ is a set of key constraints.
- Main Ingredients: Reduction to ILP + Database Techniques
- Extensive experimentation with 21 conjunctive queries for which $\textsc{Con}(q, \Sigma)$ spans all three possibilities: FO-rewritable, in P but not FO-rewritable, coNP-complete.
- Comparison of EQUIP with both ConQuer and ConsEx:
  - ConQuer does better than EQUIP on conjunctive queries for which $\textsc{Con}(q, \Sigma)$ is FO-rewritable.
  - EQUIP significantly outperforms ConsEX on conjunctive queries for which $\textsc{Con}(q, \Sigma)$ is in P but not FO-rewritable or $\textsc{Con}(q, \Sigma)$ is coNP-complete.

EQUIP vs. ConQuer



EQUIP vs. ConsEx



Queries for Evaluating EQUIP

Complexity of CQA:	coNP-complete
$Q_1() :-$	$R_5(\underline{x}, y, z), R_6(\underline{x'}, y, w)$
$Q_2(z) :-$	$R_5(\underline{x}, y, z), R_6(\underline{x'}, y, w)$

Complexity of CQA:	in P, not FO-rewritable
$Q_8() :-$	$R_3(\underline{x}, y, z), R_4(\underline{y}, x, w)$
$Q_9(z) :-$	$R_3(\underline{x}, y, z), R_4(\underline{y}, x, w)$

Complexity of CQA:	FO-rewritable
$Q_{15}(z) :-$	$R_1(\underline{x}, y, z), R_2(\underline{y}, v, w)$
$Q_{16}(z, w) :-$	$R_1(\underline{x}, y, z), R_2(\underline{y}, v, w)$
$Q_{17}(z) :-$	$R_1(\underline{x}, y, z), R_2(\underline{y}, v), R_7(\underline{v}, u, d)$
$Q_{18}(z, w) :-$	$R_1(\underline{x}, y, z), R_2(\underline{y}, v), R_7(\underline{v}, u, d)$

A Vision for a Comprehensive CQA System

- Preprocessing: Use the complexity classification of $\textsc{Cqa}(q, \Sigma)$ to determine the evaluation strategy.
  - Given q and Σ, determine if $\textsc{Cqa}(q, \Sigma)$ is FO-rewritable or in P but not FO-rewritable or coNP-complete.
- Module A: $\textsc{Cqa}(q, \Sigma)$ is FO-rewritable
  FO-rewriting algorithm + Database Engine
- Module B: $\textsc{Cqa}(q, \Sigma)$ is in P but not FO-rewritable
  Direct Algorithm or
  Reduction to Linear Programming + LP Solver
- Module C: $\textsc{Cqa}(q, \Sigma)$ is 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.