Logic and Databases

Phokion G. Kolaitis

UC Santa Cruz & IBM Research - Almaden

Lecture 5

Alternative Semantics of Queries

- Bag Semantics

  We focused on the containment problem for conjunctive queries under bag semantics.

- Probabilistic Databases

  We focused on the data complexity of conjunctive query on tuple independence databases

  Today, we will discuss

- Inconsistent Databases

  The focus will be on the data complexity of conjunctive queries in this framework.

Logic and Databases

- 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.
- So far, we have discussed the use of logic as a database query language.
- In what follows, we will discuss some aspects of the use of logic as a specification language to express integrity constraints.

Integrity Constraints in Databases

- Integrity Constraints are semantic restrictions that the data at hand ought to obey.
- 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 (local-as-view) constraints, GAV (global-as-view) constraints.

Study of Integrity Constraints in Databases

- Initial focus on the decidability and complexity of the implication problem for integrity constraints:
  Given Σ and Σ', does $\Sigma \models \Sigma'$?
- More recent extensive study of egds and tgds in data integration and data exchange.
  They have been used to design schema-mapping languages for formalizing data inter-operability tasks.
- More recent extensive study of the decidability and complexity of query answering over inconsistent databases, i.e., databases that violate integrity constraints specified by egds and tgds.

Equality-Generating Dependencies

Definition

- Functional Dependency $R : X \to Y$
  If two tuples in R agree on X, then they agree on Y.
- Key Constraint $R : X \to Y$, where Y is the set of attributes of R that are not in X.

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

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

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).

$\textsc{Certainty}(q, \Sigma)$ is the following decision problem:
Given a database I, is $\textsc{Con}(q, I, \Sigma)$ true?
(i.e., is q true on every repair of I?)

Fact

- Repair checking is in P (in fact, it is in L).
- $\textsc{Certainty}(q, \Sigma)$ is in coNP.

Complexity of CQA: An Illustration

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

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

- 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 $\textsc{Certainty}(q_i, \Sigma)$, where $i = 1, 2, 3$?

Complexity of CQA: An Illustration

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

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

- 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 $\text{CERTAINTY}(q, \Sigma)$?

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 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:

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

Moreover, the dichotomy is effective: we can decide in PTIME whether $\textsc{Certainty}(q, \Sigma)$ 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 $\textsc{Certainty}(q, \Sigma)$

$\textsc{Certainty}(q, \Sigma)$	$\nearrow$	coNP-complete
		not coNP-complete, not in P
	$\searrow$	P

Progress towards the Dichotomy for $\textsc{Certainty}(q, \Sigma)$

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:

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

Moreover, this dichotomy is decidable in quadratic time.

Progress towards the Dichotomy for $\textsc{Certainty}(q, \Sigma)$

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:

- $\textsc{Certainty}(q, \Sigma)$ is in P.
- $\textsc{Certainty}(q, \Sigma)$ 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.

The Attack Graph

Σ a set of key constraints with one key per relation.
q a Boolean self-join free conjunctive query

- $K(q) = \{\mathrm{key}(F) \to \mathrm{Var}(F) : F \text{ is an atom of } q\}$.
- $F^{+,q} = \{x \in \mathrm{Var}(q) : K(q \setminus F) \models \mathrm{key}(F) \to x\}$.
- F attacks G, denoted $F \rightsquigarrow G$, if there is a sequence $F_1, \ldots F_n$ such that
  - $F_1 = F$ and $F_n = G$.
  - $\mathrm{Var}(F_i) \cap \mathrm{Var}(F_{i+1}) \not\subseteq F^{+,q}$, for every $i \leq n-1$,
- An attack $F \rightsquigarrow G$ is weak if $K(q) \models \mathrm{key}(F) \to \mathrm{key}(G)$; otherwise, the attack is strong.
- A cycle in the attack graph is strong if it contains at least one strong attack.

Progress towards the Dichotomy for $\textsc{Certainty}(q, \Sigma)$

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 $\textsc{Certainty}(q, \Sigma)$ is in P and, in fact, it FO-rewritable; otherwise, $\textsc{Certainty}(q, \Sigma)$ is L-hard, hence it is not FO-rewritable.
- If the attack graph contains no strong cycle, then $\textsc{Certainty}(q, \Sigma)$ is in P.
- If the attack graph contains a strong cycle, then $\textsc{Certainty}(q, \Sigma)$ 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 $\textsc{Certainty}(q, \Sigma)$ 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 $\textsc{Certainty}(q, \Sigma)$ is in P.
- If $\mathrm{key}(R) \cup \mathrm{key}(S) \not\subseteq \mathrm{Var}(R) \cap \mathrm{Var}(S)$, then $\textsc{Certainty}(q, \Sigma)$ 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 $\textsc{Certainty}(q, \Sigma)$ 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 $\textsc{Certainty}(q, \Sigma)$ is in P.
- If $\mathrm{key}(R) \cup \mathrm{key}(S) \not\subseteq \mathrm{Var}(R) \cap \mathrm{Var}(S)$, then $\textsc{Certainty}(q, \Sigma)$ is coNP-complete.

Examples:

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

Beyond the Koutris-Wijsen Dichotomy Theorem

Open Problems

- Prove the Dichotomy Conjecture for $\textsc{Certainty}(q, \Sigma)$, where Σ 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 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 $\textsc{Certainty}(q, \Sigma)$, 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 $\text{C}\textsc{ertainty}(q, \Sigma)$, 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 Constaint Satisfaction (equivalently, the query complexity of 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.

Fontaine - 2015:

Discovered an *a priori* unexpected connection between Consistent Query Answering and Constaint Satisfaction (equivalently, the query complexity of conjunctive queries).

Theorem (Fontaine - 2015)

If the dichotomy theorem holds for CERTAINTY(q, Σ), where Σ is a finite set of GAV constraints and q is a union of Boolean conjunctive queries, then the Feder-Vardi Conjecture is true, i.e., a dichotomy theorem holds for the family $P_D(CQ)$ of problems about the query complexity of CQ-evaluation.

Global-As-View (GAV) Constraints

Definition:

- Recall that a tgd is a constraint of the form
  $$\forall \mathbf{x}(\phi(\mathbf{x}) \rightarrow \exists \mathbf{y}\psi(\mathbf{x},\mathbf{y})),$$
  where $\phi(\mathbf{x})$ and $\psi(\mathbf{x},\mathbf{y})$ are conjunctions of atoms.
- A global-as-view (GAV) constraint is a tgd of the form
  $$\forall \mathbf{x}(\phi(\mathbf{x}) \rightarrow T(\mathbf{x})),$$
  where $T(\mathbf{x})$ is a single atom.
  In effect, a GAV constraint is a Horn clause.

Examples:

- $\forall x, y(R(x,y) \rightarrow R(y,x))$
- $\forall x, y, z(R(x,z) \wedge S(z,y) \rightarrow T(x,y))$

Constraint Satisfaction and CQA for GAV Constraints

Fact

If Σ is a finite set of GAV constraints and q is a union of Boolean conjunctive queries, then $\oplus$-$\textsc{Certainty}(q, \Sigma)$ is in coNP.

Constraint Satisfaction and CQA for GAV Constraints

Fact

If Σ is a finite set of GAV constraints and q is a union of Boolean conjunctive queries, then $\oplus$-$\textsc{Certainty}(q, \Sigma)$ is in coNP.

Theorem (Fontaine - 2015)

For every relational structure $\mathbf{B}$, there is a finite set Σ of GAV constraints and a union q of Boolean conjunctive queries, such that $\overline{\mathrm{CSP}(\mathbf{B})}$ is PTIME-equivalent to $\oplus$-$\textsc{Certainty}(q, \Sigma)$.

Corollary

If the dichotomy theorem holds for $\oplus$-$\textsc{Certainty}(q, \Sigma)$, where Σ is a finite set of GAV constraints and q is a union of Boolean conjunctive queries, then the dichotomy theorem holds for $\mathrm{CSP}(\mathbf{B})$, 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 $f \succ g$.

$J \setminus K$	f
$J \cap K$	
$K \setminus J$	g

$f \succ g$

- 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 $\to$ instructor and instructor, term $\to$ 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, term $\rightarrow$ course	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?

Topics Covered

- Logic and Database Query Languages
  - Relational Algebra and Relational Calculus
  - Conjunctive Queries and their variants
  - Datalog
- Query Evaluation, Query Containment, Query Equivalence
- Other aspects of Conjunctive Query Evaluation
  - Acyclic joins, treewidth, bounds on the size of natural joins
- Alternative Semantics
  - Bag Databases, Probabilistic Databases, Inconsistent Databases
- Emphasis on the interplay between databases, logic, and computational complexity

Topics Not Covered

- Automata-theoretic techniques in databases
- Reasoning about database dependencies (the implication problem)
- Incomplete databases
- Information integration, data exchange, data warehousing
- Data privacy and security
- Data provenance

  Guest Lecture by Val Tannen
- Beyond relational databases
  - Semi-structured data and XML
  - Graph databases, web data

Logic and Databases are inextricably intertwined