Computing Query Answers with Consistent Support

Jui-Yi Kao
Stanford University

Advised by: Michael Genesereth
Inconsistency in Databases

• If the data in a database violates the applicable ICs, we say the data is inconsistent.

• Care must be taken to avoid nonsensical answers e.g. Julius Caesar born twice!

<table>
<thead>
<tr>
<th>Birth Year:</th>
<th>person</th>
<th>date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Julius Caesar</td>
<td>100 BC</td>
<td></td>
</tr>
<tr>
<td>Julius Caesar</td>
<td>102 BC</td>
<td></td>
</tr>
<tr>
<td>Edgar Codd</td>
<td>1923 AD</td>
<td></td>
</tr>
</tbody>
</table>

IC:
Each person a unique birth year
Why inconsistencies?

• integration of autonomous data sources.
  – Two sources of data may show two surnames for the same person because
    • the two sources are out of sync
    • or one was incorrectly entered.
  – two data sources may claim two different birth years for Julius Caesar.

• unenforced constraints.
  – legacy system
  – efficiency
  – unsupported types

• preservation of information
Consistent Support

• many methods proposed for querying inconsistent data
• we do EE
• motivate with pqr example
• define EE
### Example - Data

<table>
<thead>
<tr>
<th>institution &lt;student, inst&gt;</th>
<th>degree &lt;student, degree&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>(id1, &quot;Stanford University&quot;)</td>
<td>(id1, &quot;MA&quot;)</td>
</tr>
<tr>
<td>(id2, &quot;Academy of Art&quot;)</td>
<td>(id2, &quot;MS&quot;)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>dept &lt;student, dept&gt;</th>
<th>ca_institution &lt;inst&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>(id1, cs)</td>
<td>(&quot;Stanford University&quot;)</td>
</tr>
<tr>
<td>(id2, cs)</td>
<td>(&quot;Academy of Art&quot;)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>name &lt;student, name&gt;</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(id1, &quot;Alyssa&quot;)</td>
<td></td>
</tr>
<tr>
<td>(id2, &quot;Alyssa&quot;)</td>
<td>(&quot;Santa Clara University&quot;)</td>
</tr>
<tr>
<td></td>
<td>(&quot;San Jose State&quot;)</td>
</tr>
</tbody>
</table>
Constraint

institution <student, inst>
(id1, "Stanford University")
(id2, "Academy of Art")

department <student, dept>
(id1, cs)
(id2, cs)

degree <student, degree>
(id1, "MA")
(id2, "MS")

canada_institution <inst>
("Stanford University")
("Academy of Art")
("Santa Clara University")
("San Jose State")

name<student, name>
(id1, "Alyssa")
(id2, "Alyssa")

Constraint (1)
• institution(X,"Stanford University") \land department(X,"Computer Science") \rightarrow \neg degree(X,"MA")
### Constraint

<table>
<thead>
<tr>
<th>institution &lt;student, inst&gt;</th>
<th>degree &lt;student, degree&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>(id1, &quot;Stanford University&quot;)</td>
<td>(id1, &quot;MA&quot;)</td>
</tr>
<tr>
<td>(id2, &quot;Academy of Art&quot;)</td>
<td>(id2, &quot;MS&quot;)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>dept &lt;student, dept&gt;</th>
<th>ca_institution &lt;inst&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>(id1, cs)</td>
<td>(&quot;Stanford University&quot;)</td>
</tr>
<tr>
<td>(id2, cs)</td>
<td>(&quot;Academy of Art&quot;)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>name &lt;student, name&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>(id1, &quot;Alyssa&quot;)</td>
</tr>
<tr>
<td>(id2, &quot;Alyssa&quot;)</td>
</tr>
</tbody>
</table>

**Constraint (2)**
- Institution(X,"Academy of Art University") \(\rightarrow \neg\) Department(X,"Computer Science")
### Answer

<table>
<thead>
<tr>
<th>institution &lt;student, institution&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>(id1, &quot;Stanford University&quot;)</td>
</tr>
<tr>
<td>(id2, &quot;Academy of Art University&quot;)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>department &lt;student, dept&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>(id1, &quot;Computer Science&quot;)</td>
</tr>
<tr>
<td>(id2, &quot;Computer Science&quot;)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>name &lt;student, name&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>(id1, &quot;Alyssa&quot;)</td>
</tr>
<tr>
<td>(id2, &quot;Alyssa&quot;)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>degree &lt;student, degree&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>(id1, &quot;MA&quot;)</td>
</tr>
<tr>
<td>(id2, &quot;MS&quot;)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bayarea_institution &lt;institution&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>(&quot;Stanford University&quot;)</td>
</tr>
<tr>
<td>(&quot;Academy of Art University&quot;)</td>
</tr>
<tr>
<td>(&quot;Santa Clara University&quot;)</td>
</tr>
<tr>
<td>(&quot;San Jose State University&quot;)</td>
</tr>
</tbody>
</table>

- answers(X) :- inst(X, Y), caInst(Y), dept(X, cs), name(X, alyssa)

- answers(id1)
<table>
<thead>
<tr>
<th>Institution</th>
<th>Degree</th>
<th>Department</th>
<th>Bay Area Institution</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Stanford University&quot;</td>
<td>&quot;MA&quot;</td>
<td>&quot;Computer Science&quot;</td>
<td>&quot;Stanford University&quot;</td>
</tr>
<tr>
<td>&quot;Academy of Art University&quot;</td>
<td>&quot;MS&quot;</td>
<td>&quot;Computer Science&quot;</td>
<td>&quot;Academy of Art University&quot;</td>
</tr>
<tr>
<td>&quot;Santa Clara University&quot;</td>
<td></td>
<td></td>
<td>&quot;Santa Clara University&quot;</td>
</tr>
<tr>
<td>&quot;San Jose State University&quot;</td>
<td></td>
<td></td>
<td>&quot;San Jose State University&quot;</td>
</tr>
</tbody>
</table>

- `answers(X) :- inst(X, Y), caInst(Y), dept(X, cs), name(X, alyssa)`
- `id2 is not an answer!`
Naïve Method

• Consider each consistent (maximal) subset of the data
• Find the the standard query answers on each subset
• Problem: There may be exponentially many consistent maximal subsets!

\[ p(A,B) \]

<table>
<thead>
<tr>
<th></th>
<th>a1</th>
<th>a1</th>
<th>a2</th>
<th>a2</th>
<th>...</th>
<th>an</th>
<th>an</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>b0</td>
<td>b1</td>
<td>b0</td>
<td>b1</td>
<td>...</td>
<td>b0</td>
<td>b1</td>
</tr>
</tbody>
</table>

FD: \[ A \rightarrow B \]

A relation of \( 2^n \) tuples has \( 2^n \) consistent maximal subsets!
A Rewriting Approach

$B \models C : \text{Constraints}$

$Q : \text{Original}$

$Q' : \text{Rewritten}$

if and only if

$B : \text{Database instance}$

$B \models Q(a)$

$B \models Q'(a)$
A Rewriting Approach

- Given query Q and constraints C
- Rewrite Q as Q' so that for any database instance B:
  the strict entailment answers according to Q is exactly the standard answers according to Q'
  \[ B \models_{C} Q(a) \iff B \models Q'(a) \]

- Polynomial data complexity for first-order query
- Leverage standard database technologies and techniques to evaluate Q'
Setting

• Constraints:
  – Function-free
  – Universal clauses (no existential quantifier)
  – Finite closure under resolution

• Queries:
  – First-order queries, equivalently:
    • Relational Algebra
    • Relational Calculus
    • Nonrecursive-Datalog

• Database:
  – Closed World Assumption
Rewriting Algorithm

• Close constraints under resolution
• Write query body as unit clauses (b-clauses)
  – institution(X, Y)
  – bayarea_institution(Y)
  – department(X, "Computer Science")
  – name(X, "Alyssa")
• Apply unit resolutions between b-clauses and constraints. Each sequence of units resolutions that leads to an empty clause is a variable binding of the query body that violates the constraints
  • Rewrite within inequalities to prevent
Rewriting Examples

• \( q(X) :\) inst(X,Y), caInst(Y), dept(X,cs), name(X, alyssa)  
  \[ \text{rewriting} \]

• \( q'(X) :\) inst(X,Y), caInst(Y), dept(X,cs), name(X, alyssa)  
  \[ Y \neq \text{art} \]
Blocking Inconsistent Data

• Given:
  – Datalog rule: \( p(X) :- \varphi(X,Y) \)
  – constraint clause \( c \)

• Determine:
  – Which data bindings \( \sigma \) make \( \varphi(X,Y)\sigma \) violates clause \( c \)?

• Solution:
  – \( \varphi(X,Y)\sigma \) violates \( c \) \iff \( d \) subsumes \( \neg \varphi(X,Y)\sigma \)
Blocking Inconsistent Data

\[ \neg \text{dept}(X,cs) \lor \neg \text{degree}(X,\text{ma}) \]
\[ \neg \text{inst}(X,\text{art}) \lor \neg \text{dept}(X,cs) \]

Closed under resolution

\[ q(X) :- \text{inst}(X,Y),\text{calInst}(Y),\text{dept}(X,cs), \text{name}(X,\text{alyssa}) \]

\[ \text{inst}(X,Y), \text{calInst}(Y), \text{dept}(X,cs), \text{name}(X,\text{alyssa}) \]
Rewriting Algorithm

- **Clauses:**
  - inst(X, Y)
  - ca_inst(Y)
  - dept(X, cs)
  - name(X, "Alyssa")
  - ¬dept(X, cs) ∨ ¬degree(X, ma) (1)
  - ¬inst(X, art) ∨ ¬dept(X, cs) (2)

- Y ← art
- Y ≠ art
•answers'(X) :- institution(X, Y),
bayarea_institution(Y),
department(X, "Computer Science"),
name(X, "Alyssa"),
Y != "Academy of Arts University"

•answers'(id1)
<table>
<thead>
<tr>
<th>institution &lt;student, institution&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>(id1, &quot;Stanford University&quot;)</td>
</tr>
<tr>
<td>(id2, &quot;Academy of Art University&quot;)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>department &lt;student, dept&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>(id1, &quot;Computer Science&quot;)</td>
</tr>
<tr>
<td>(id2, &quot;Computer Science&quot;)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>name&lt;student, name&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>(id1, &quot;Alyssa&quot;)</td>
</tr>
<tr>
<td>(id2, &quot;Alyssa&quot;)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>degree &lt;student, degree&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>(id1, &quot;MA&quot;)</td>
</tr>
<tr>
<td>(id2, &quot;MS&quot;)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bayarea_institution &lt;institution&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>(&quot;Stanford University&quot;)</td>
</tr>
<tr>
<td>(&quot;Academy of Art University&quot;)</td>
</tr>
<tr>
<td>(&quot;Santa Clara University&quot;)</td>
</tr>
<tr>
<td>(&quot;San Jose State University&quot;)</td>
</tr>
</tbody>
</table>

• `answers'(X) :- institution(X, Y), bayarea_institution(Y), department(X, "Computer Science"), name(X, "Alyssa"), Y != "Academy of Arts University"

• `answer'(id2) is blocked`
Features

- Polynomial data complexity

- The query rewriting is done once and may be evaluated on changing data

- Standard techniques apply to rewritten query e.g.,
  - query planning
  - differential view maintenance
  - distributed query evaluation
Limitations

- Universal clauses express typical classes integrity constraints:
  - functional dependencies
  - denial constraints
  - etc.
- Cannot express referential integrity constraints
  - lacks existential quantification
TODO: Query and Constraint Classes

• finding answers under broader classes of constraints
  – General first-order constraints
  – built-in predicates beyond =

• finding answers to broader classes of queries
  – recursive queries
  – aggregates

• Ideas:
  – careful skolemization
  – control resolution
  – interaction between constraint type and query type
TODO: Stop Any Time

• Resolution closure may not terminate or may take a long time

• Idea: augment the query as resolution takes place

• Then the procedure can be stopped at any time and the most complete rewriting computed so far is returned
TODO: View Maintenance

- Often, a query is not evaluated just once. Instead, a view is maintained.
- E.g., maintain list of emails of Bay Area CS students resulting from a query
- View can be updated "differentially" based on changes to the underlying data
- Investigate adapting and applying existing differential view maintenance techniques in the presence of inconsistencies
- Investigate algorithms and analyze complexity
• TODO: others

• change constraints

• distributed query evaluation
  – In a federated databases setting, it is desirable to distribute the work of query evaluation.
  – push work down to data sources

• "local" constraints and "global" constraints

• Take or leave each data source in its entirety
Applications

• Querying and updating federated autonomous databases
  – use strict entailment to find consistently supported consequences or update propagations
• Update through view
  – a change to view often cannot be uniquely resolved into changes to base relations
  – change the materialized view nonetheless and then draw consistently supported conclusions
• "Collaborative Data Management"
• Logical spreadsheets
Prior Work

• Consistent Query Answers
  – (Arenas, Bertossi, Chomicki 98) (Fuxman & Miller 07) (Chomicki & Marcinkowski 04) (Bertossi 06)

• Argumentation
  – (Elvang-Goransson & Hunter 95) (Efstathiou & Hunter 08) (Besnard & Hunter 06) (Besnard & Hunter 05)

• Logical spreadsheets
  – (Kassoff & Genesereth 07)

• Possibilistic databases
  – (Pradhan 03) (Pradhan 05)
Thank you

• Questions
• Comments
• Suggestions
• Advice
Computing Query Answers with Consistent Support

Jui-Yi Kao
Stanford University

Advised by: Michael Genesereth
Strict Existential Entailment

- Developed by Kassoff and Genesereth for Logical Spreadsheets
- An answer is strictly existentially entailed iff it is supported by a consistent subset of the data
- Given a database instance $B$ and constraint rules $C$, an answer $a$ is strictly existentially entailed iff there is a subset $B'$ of $B$ such that $B' \cup \{C\} \not\models \perp$ and $B' \cup \{C\} \vdash a$
- Finding all strictly existentially entailed answers to a query solves the problem on the previous slide.
Querying and updating inconsistent data

• When a system integrates data from independent databases global constraints are often violated.
  – Find-a-classmate search
  – Product search
• How to query the data in the case of inconsistencies?
• If the system links independent databases, how can a change to one update the others?
  – Changing customer information
  – Changing information on social networks
The Problem

- Given a database that is (possibly) inconsistent with the integrity constraints,
- We can view answering a query as making an argument for an answer using the facts in the data.
- Standard query semantics asks for all query answers supported by an argument.
- But an argument that violates the ICs is clearly incorrect.
- Our goal is to find all query answers which are supported by an argument consistent w.r.t the ICs.
Related work

- Data integration
- Data warehousing / cleaning
- Update through views
- Probabilistic / uncertain databases
- Lineage and provenance
- Consistent query answers
- I would like to contribute mainly in update and query using credulous semantics

- composing credulous answers
  - conditional answers
- disjunctive information