Finding Consistent Answers from Inconsistent Data: Systems, Algorithms, and Complexity

Xiating Ouyang

University of Wisconsin–Madison

PhD Defense, November 21, 2023

Committee: Uri Andrews, Jin-Yi Cai, Paris Koutris, Jignesh Patel, Jef Wijsen
Finding Consistent Answers from Inconsistent Data: Systems, Algorithms, and Complexity
JZ: want to go biking today at 6pm?
XO: ... is that a good idea?
JZ: that's not what I see ... 
Us: let's play badminton instead ...

Xiating Ouyang
Consistent Query Answering
PhD Defense 3 / 71
JZ: want to go biking today at 6pm?

. . . is that a good idea?

Us: let's play badminton instead . . .
JZ: want to go biking today at 6pm?

XO: ...is that a good idea?
JZ: want to go biking today at 6pm?

XO: ...is that a good idea?

JZ: that's not what I see ...
JZ: want to go biking today at 6pm?

XO: ... is that a good idea?

JZ: that’s not what I see . . .

Us: let’s play badminton instead . . .
Alternatives from NLP, ML models . . .

Our focus: relational databases

Model guidance only. Expert interpretation required. Check NHC/CPC/JTWC official forecasts.
- Alternatives from NLP, ML models . . .
- Our focus: relational databases
<table>
<thead>
<tr>
<th>City</th>
<th>Weather</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>* MSN</td>
<td>Rainy</td>
<td></td>
</tr>
<tr>
<td>* MSN</td>
<td>Sunny</td>
<td></td>
</tr>
<tr>
<td>LA</td>
<td>Sunny</td>
<td></td>
</tr>
<tr>
<td>Seattle</td>
<td>Rainy</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Weather</th>
<th>Biking</th>
<th>Badmin.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rainy</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Sunny</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>−37 deg.</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

- Inconsistent data: data that violates integrity constraints
- Primary key (PK) constraint: ≤ 1 tuple for each PK value
### Forecast

<table>
<thead>
<tr>
<th>City</th>
<th>Weather</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSN</td>
<td>Rainy</td>
</tr>
<tr>
<td>MSN</td>
<td>Sunny</td>
</tr>
<tr>
<td>LA</td>
<td>Sunny</td>
</tr>
<tr>
<td>Seattle</td>
<td>Rainy</td>
</tr>
</tbody>
</table>

### Activity

<table>
<thead>
<tr>
<th>Weather</th>
<th>Biking</th>
<th>Badmin.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rainy</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Sunny</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>−37 deg.</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

- **Inconsistent data**: data that violates integrity constraints
- Primary key (PK) constraint: \( \leq 1 \) tuple for each **PK value**
Primary key constraint (violated)

- Metadata of stackoverflow.com as of 02/2021 from Stack Exchange Data Dump
- 551M rows, ~400 GB

<table>
<thead>
<tr>
<th>Table</th>
<th># of rows</th>
<th>inconsistencyRatio</th>
<th>blockSize</th>
<th># of Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Users</td>
<td>14M</td>
<td>0%</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>Posts</td>
<td>53M</td>
<td>0%</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>PostHistory</td>
<td>141M</td>
<td>0.001%</td>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>Badges</td>
<td>40M</td>
<td>0.58%</td>
<td>941</td>
<td>4</td>
</tr>
<tr>
<td>Votes</td>
<td>213M</td>
<td>30.9%</td>
<td>1441</td>
<td>6</td>
</tr>
</tbody>
</table>

\[
\text{inconsistencyRatio} = \frac{\# \text{ facts violating PK constraint}}{\# \text{ of rows}}
\]
\[
\text{blockSize} = \text{max. } \# \text{ facts with the same PK}
\]
Finding Consistent Answers from Inconsistent Data: Systems, Algorithms, and Complexity
Finding consistent answers

<table>
<thead>
<tr>
<th>Forecast</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>City</strong></td>
<td><strong>Weather</strong></td>
</tr>
<tr>
<td>* MSN</td>
<td>Rainy</td>
</tr>
<tr>
<td>* MSN</td>
<td>Sunny</td>
</tr>
<tr>
<td>LA</td>
<td>Sunny</td>
</tr>
<tr>
<td>Seattle</td>
<td>Rainy</td>
</tr>
</tbody>
</table>

$q$: find all cities that are suitable for badminton at 6pm

```sql
SELECT DISTINCT city
FROM Forecast, Activity
WHERE Forecast.weather = Activity.weather
AND Badminton. = "Yes"
```

$q(x) = \exists y, z : \text{Forecast}(x, y) \land \text{Activity}(y, z, "Yes")$

$q(db) = \{\text{Answers of } q \text{ on } db\}
= \{\text{city} \mid q_{[x \rightarrow \text{city}]} \text{ is true on } db\}
= \{\text{city} \mid db \models q_{[x \rightarrow \text{city}]}\}
= \{MSN, LA, Seattle\}$
## Finding consistent answers

<table>
<thead>
<tr>
<th>City</th>
<th>Weather</th>
<th>Activity</th>
<th>Weather</th>
<th>Biking</th>
<th>Badminton</th>
</tr>
</thead>
<tbody>
<tr>
<td>* MSN</td>
<td>Rainy</td>
<td></td>
<td>Rainy</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>* MSN</td>
<td>Sunny</td>
<td></td>
<td>Sunny</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>LA</td>
<td>Sunny</td>
<td></td>
<td>−37 deg.</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Seattle</td>
<td>Rainy</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\( q \): find all cities that are suitable for badminton at 6pm

```sql
SELECT DISTINCT city
FROM Forecast, Activity
WHERE Forecast.weather = Activity.weather
AND Badminton. = "Yes"
```

\( q(x) = \exists y, z : \text{Forecast}(x, y) \land \text{Activity}(y, z, "Yes") \)

\( q(db) = \{ \text{Answers of } q \text{ on } db \} \)

\( = \{ \text{city} | q[x \rightarrow \text{city}] \text{ is true on } db \} \)

\( = \{ \text{city} | db \models q[x \rightarrow \text{city}] \} \)

\( = \{ \text{MSN, LA, Seattle} \} \)
Finding consistent answers

<table>
<thead>
<tr>
<th>City</th>
<th>Weather</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSN</td>
<td>Rainy</td>
</tr>
<tr>
<td>MSN</td>
<td>Sunny</td>
</tr>
<tr>
<td>LA</td>
<td>Sunny</td>
</tr>
<tr>
<td>Seattle</td>
<td>Rainy</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Weather</th>
<th>Biking</th>
<th>Badminton.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rainy</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Sunny</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>-37 deg.</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

$q$: find all cities that are suitable for badminton at 6pm

```
SELECT DISTINCT city
FROM Forecast, Activity
WHERE Forecast.weather = Activity.weather
    AND Badminton. = "Yes"
```

$q(x) = \exists y, z : Forecast(x, y) \land Activity(y, z, "Yes")$

$q(db) = \{\text{Answers of } q \text{ on } db\}$

$= \{\text{city} \mid q_{\{x\rightarrow \text{city}\}} \text{ is true on } db\}$

$= \{\text{city} \mid db \models q_{\{x\rightarrow \text{city}\}}\}$

$= \{\text{MSN, LA, Seattle}\}$
Finding consistent answers

<table>
<thead>
<tr>
<th>City</th>
<th>Weather</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>* MSN</td>
<td>Rainy</td>
<td>Rainy</td>
</tr>
<tr>
<td>* MSN</td>
<td>Sunny</td>
<td>Sunny</td>
</tr>
<tr>
<td>LA</td>
<td>Sunny</td>
<td>Yes</td>
</tr>
<tr>
<td>Seattle</td>
<td>Rainy</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>−37 deg.</td>
<td>No</td>
</tr>
</tbody>
</table>

$q$: find all cities that are suitable for badminton at 6pm

```
SELECT DISTINCT city
FROM Forecast, Activity
WHERE Forecast.weather = Activity.weather
AND Activity.weather = "Yes"
```

$q(x) = \exists y, z : Forecast(x, y) \land Activity(y, z, "Yes")$

$q(db) = \{\text{Answers of } q \text{ on } db\}$

$= \{\text{city} \mid q_{x \rightarrow \text{city}} \text{ is true on } db\}$

$= \{\text{city} \mid db \models q_{x \rightarrow \text{city}}\}$

$= \{\text{MSN, LA, Seattle}\}$
### Finding consistent answers

<table>
<thead>
<tr>
<th>Forecast</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>City</strong></td>
<td><strong>Weather</strong></td>
</tr>
<tr>
<td>* MSN</td>
<td>Rainy</td>
</tr>
<tr>
<td>* MSN</td>
<td>Sunny</td>
</tr>
<tr>
<td>LA</td>
<td>Sunny</td>
</tr>
<tr>
<td>Seattle</td>
<td>Rainy</td>
</tr>
</tbody>
</table>

**q**: find all cities that are suitable for badminton at 6pm

```
SELECT DISTINCT city
FROM Forecast, Activity
WHERE Forecast.weather = Activity.weather
AND Badminton. = "Yes"
```

\[
q(x) = \exists y, z: \text{Forecast}(x, y) \land \text{Activity}(y, z, "Yes")
\]

\[
q(db) = \{\text{Answers of } q \text{ on } db\}
\]

\[
= \{\text{city} \mid q_{[x\rightarrow \text{city}]} \text{ is true on db}\}
\]

\[
= \{\text{city} \mid db \models q_{[x\rightarrow \text{city}]}\}
\]

\[
= \{\text{MSN, LA, Seattle}\}
\]
So that we are on the same page...

<table>
<thead>
<tr>
<th>DB system</th>
<th>DB theory</th>
<th>Logic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Finite relations</td>
<td>Finite structure w/o func.</td>
</tr>
<tr>
<td>SQL Query w/o Aggr.</td>
<td>Query</td>
<td>First-order formula</td>
</tr>
<tr>
<td>Sel.-Proj.-Join Query</td>
<td>Conjunctive query (CQ)</td>
<td>Formula in $\text{FO} (\exists, \land)$</td>
</tr>
</tbody>
</table>
Finding consistent answers

<table>
<thead>
<tr>
<th>Forecast</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>City</strong></td>
<td><strong>Weather</strong></td>
</tr>
<tr>
<td>* MSN</td>
<td>Rainy</td>
</tr>
<tr>
<td>* MSN</td>
<td>Sunny</td>
</tr>
<tr>
<td>LA</td>
<td>Sunny</td>
</tr>
<tr>
<td>Seattle</td>
<td>Rainy</td>
</tr>
</tbody>
</table>

$q$: find all cities that are suitable for badminton at 6pm

$q(db) = \{\text{MSN, LA, Seattle}\} \ldots$ on dirty data

Data cleaning
Finding consistent answers

<table>
<thead>
<tr>
<th>City</th>
<th>Weather</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>* MSN</td>
<td>Rainy</td>
<td></td>
</tr>
<tr>
<td>* MSN</td>
<td>Sunny</td>
<td></td>
</tr>
<tr>
<td>LA</td>
<td>Sunny</td>
<td></td>
</tr>
<tr>
<td>Seattle</td>
<td>Rainy</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Weather</th>
<th>Biking</th>
<th>Badminton</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rainy</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Sunny</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>−37 deg.</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

$q$: find all cities that are suitable for badminton at 6pm

$q(db) = \{MSN, LA, Seattle\} \ldots$ on dirty data

Data cleaning
Finding consistent answers

<table>
<thead>
<tr>
<th>Forecast</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>City</td>
<td>Weather</td>
</tr>
<tr>
<td>* MSN</td>
<td>Rainy</td>
</tr>
<tr>
<td>* MSN</td>
<td>Sunny</td>
</tr>
<tr>
<td>LA</td>
<td>Sunny</td>
</tr>
<tr>
<td>Seattle</td>
<td>Rainy</td>
</tr>
</tbody>
</table>

$q$: find all cities that are suitable for badminton at 6pm

$q(db) = \{MSN, LA, Seattle\} \ldots$ on dirty data

Data cleaning

$q(rep)$
Finding consistent answers

<table>
<thead>
<tr>
<th>City</th>
<th>Weather</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>* MSN</td>
<td>Rainy</td>
<td></td>
</tr>
<tr>
<td>* MSN</td>
<td>Sunny</td>
<td></td>
</tr>
<tr>
<td>LA</td>
<td>Sunny</td>
<td></td>
</tr>
<tr>
<td>Seattle</td>
<td>Rainy</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Weather</th>
<th>Biking</th>
<th>Badmin.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rainy</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Sunny</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>−37 deg.</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

$q$: find all cities that are suitable for badminton at 6pm

$q(db) = \{\text{MSN, LA, Seattle}\} \ldots$ on dirty data

Data cleaning: 2 repairs

$q(rep) \; \text{vs.} \; q(rep')$
Finding consistent answers

Forecast

<table>
<thead>
<tr>
<th>City</th>
<th>Weather</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSN</td>
<td>Rainy</td>
</tr>
</tbody>
</table>
* MSN   | Sunny   |
| LA     | Sunny   |
| Seattle| Rainy   |

Activity

<table>
<thead>
<tr>
<th>Weather</th>
<th>Biking</th>
<th>Badminton</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rainy</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Sunny</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>−37 deg.</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

q: find all cities that are suitable for badminton at 6pm

\[ q(db) = \{MSN, LA, Seattle\} \ldots \text{on dirty data} \]

Data cleaning: 2 repairs (can be exponential...)
Finding consistent answers

<table>
<thead>
<tr>
<th>Forecast</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>City</td>
<td>Weather</td>
</tr>
<tr>
<td>* MSN</td>
<td>Rainy</td>
</tr>
<tr>
<td>* MSN</td>
<td>Sunny</td>
</tr>
<tr>
<td>LA</td>
<td>Sunny</td>
</tr>
<tr>
<td>Seattle</td>
<td>Rainy</td>
</tr>
</tbody>
</table>

$q$: find all cities that are suitable for badminton at 6pm

$q(db) = \{\text{MSN, LA, Seattle}\}$ … on dirty data

Data cleaning: 2 repairs (can be exponential…)

\[q(rep)\]
Finding consistent answers

Forecast

<table>
<thead>
<tr>
<th>City</th>
<th>Weather</th>
</tr>
</thead>
<tbody>
<tr>
<td>* MSN</td>
<td>Rainy</td>
</tr>
<tr>
<td>* MSN</td>
<td>Sunny</td>
</tr>
<tr>
<td>LA</td>
<td>Sunny</td>
</tr>
<tr>
<td>Seattle</td>
<td>Rainy</td>
</tr>
</tbody>
</table>

Activity

<table>
<thead>
<tr>
<th>Weather</th>
<th>Biking</th>
<th>Badminton</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rainy</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Sunny</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>−37 deg.</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

$q$: find all cities that are suitable for badminton at 6pm

$q(db) = \{MSN, LA, Seattle\} \ldots$ on dirty data

Data cleaning: 2 repairs (can be exponential\ldots)

Which answers are guaranteed to be returned on all repairs of dirty data?

$q(rep)$
### Finding consistent answers

<table>
<thead>
<tr>
<th>City</th>
<th>Weather</th>
<th>Forecast</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>* MSN</td>
<td>Rainy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>* MSN</td>
<td>Sunny</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LA</td>
<td>Sunny</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Seattle</td>
<td>Rainy</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Weather</th>
<th>Biking</th>
<th>Badminton</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rainy</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Sunny</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>−37 deg.</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

\[ q: \text{find all cities that are suitable for badminton at 6pm} \]

\[ q(\text{db}) = \{\text{MSN, LA, Seattle}\} \ldots \text{on dirty data} \]

Data cleaning: 2 repairs (can be exponential...)

Which answers are guaranteed to be returned on all repairs of dirty data?

\[ \bigcap \{q(\text{rep}) \mid \text{rep is a repair of db} \} \]
### Finding consistent answers

<table>
<thead>
<tr>
<th>City</th>
<th>Weather</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>* MSN</td>
<td>Rainy</td>
<td>Weather</td>
</tr>
<tr>
<td>* MSN</td>
<td>Sunny</td>
<td>Sunny</td>
</tr>
<tr>
<td>LA</td>
<td>Sunny</td>
<td>Yes</td>
</tr>
<tr>
<td>Seattle</td>
<td>Rainy</td>
<td>No</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Weather</th>
<th>Biking</th>
<th>Badminton</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rainy</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Sunny</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>−37 deg.</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

**q**: find all cities that are suitable for badminton at 6pm

\[ q(db) = \{ MSN, LA, Seattle \} \ldots \text{ on dirty data} \]

Data cleaning: 2 repairs  

(can be exponential\ldots)

Which answers are guaranteed to be returned on all repairs of dirty data?

\[ \bigcap_{rep \text{ is a repair of } db} q(rep) = \{ MSN, LA, Seattle \} \]
### Finding consistent answers

<table>
<thead>
<tr>
<th>Forecast</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>City</strong></td>
<td><strong>Weather</strong></td>
</tr>
<tr>
<td>* MSN</td>
<td>Rainy</td>
</tr>
<tr>
<td>* MSN</td>
<td>Sunny</td>
</tr>
<tr>
<td>LA</td>
<td>Sunny</td>
</tr>
<tr>
<td>Seattle</td>
<td>Rainy</td>
</tr>
</tbody>
</table>

$q$: find all cities that are suitable for badminton at 6pm

$$q(db) = \{\text{MSN, LA, Seattle}\} \ldots \text{on dirty data}$$

Data cleaning: 2 repairs

(can be exponential...)

Which answers are guaranteed to be returned on all repairs of dirty data?

**Consistent Answer** of $q$ over $db = \bigcap_{\text{rep is a repair of } db} q(\text{rep}) = \{\text{MSN, LA, Seattle}\}$
Finding consistent answers without enumeration

**Forecast**

<table>
<thead>
<tr>
<th>City</th>
<th>Weather</th>
</tr>
</thead>
<tbody>
<tr>
<td>* MSN</td>
<td>Rainy</td>
</tr>
<tr>
<td>* MSN</td>
<td>Sunny</td>
</tr>
<tr>
<td>LA</td>
<td>Sunny</td>
</tr>
<tr>
<td>Seattle</td>
<td>Rainy</td>
</tr>
</tbody>
</table>

**Activity**

<table>
<thead>
<tr>
<th>Weather</th>
<th>Biking</th>
<th>Badmin.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rainy</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Sunny</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>−37 deg.</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

**Query**

$q$: find all cities that are suitable for badminton at 6pm

$q'$: find all cities that are suitable for badminton at 6pm for all possible weather for the same city

```
SELECT DISTINCT city
FROM Forecast, Activity
WHERE Forecast.weather = Activity.weather
AND (for all weather with the same Forecast.city,
    Badmin. = "Yes")
```

$q'(x) = \exists y : \text{Forecast}(x, y) \land \forall y : (\text{Forecast}(x, y) \rightarrow \exists z : \text{Activity}(y, z, "Yes"))$
Finding consistent answers without enumeration

<table>
<thead>
<tr>
<th>City</th>
<th>Weather</th>
</tr>
</thead>
<tbody>
<tr>
<td>* MSN</td>
<td>Rainy</td>
</tr>
<tr>
<td>* MSN</td>
<td>Sunny</td>
</tr>
<tr>
<td>LA</td>
<td>Sunny</td>
</tr>
<tr>
<td>Seattle</td>
<td>Rainy</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Weather</th>
<th>Biking</th>
<th>Badminton</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rainy</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Sunny</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>−37 deg.</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

$q$: find all cities that are suitable for badminton at 6pm

$q'$: find all cities that are suitable for badminton at 6pm for all possible weather for the same city

SELECT DISTINCT city
FROM Forecast, Activity
WHERE Forecast.weather = Activity.weather
AND (for all weather with the same Forecast.city, Badmin. = "Yes")

$q'(x) = \exists y: \text{Forecast}(x, y) \land \forall y: (\text{Forecast}(x, y) \rightarrow \exists z: \text{Activity}(y, z, "Yes"))$
Finding consistent answers without enumeration

<table>
<thead>
<tr>
<th>City</th>
<th>Weather</th>
<th>Weather</th>
<th>Biking</th>
<th>Badmin.</th>
</tr>
</thead>
<tbody>
<tr>
<td>* MSN</td>
<td>Rainy</td>
<td>Rainy</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>* MSN</td>
<td>Sunny</td>
<td>Sunny</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>LA</td>
<td>Sunny</td>
<td>−37 deg.</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Seattle</td>
<td>Rainy</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$q$: find all cities that are suitable for badminton at 6pm

$q'$: find all cities that are suitable for badminton at 6pm for all possible weather for the same city

SELECT DISTINCT city
FROM Forecast, Activity
WHERE Forecast.weather = Activity.weather
AND (for all weather with the same Forecast.city, Badmin. = "Yes")

$q'(x) = \exists y: \text{Forecast}(x, y) \land \forall y: (\text{Forecast}(x, y) \rightarrow \exists z: \text{Activity}(y, z, "Yes"))$
Finding consistent answers without enumeration

<table>
<thead>
<tr>
<th>Forecast</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>City</strong></td>
<td><strong>Weather</strong></td>
</tr>
<tr>
<td>* MSN</td>
<td>Rainy</td>
</tr>
<tr>
<td>* MSN</td>
<td>Sunny</td>
</tr>
<tr>
<td>LA</td>
<td>Sunny</td>
</tr>
<tr>
<td>Seattle</td>
<td>Rainy</td>
</tr>
</tbody>
</table>

$q$: find all cities that are suitable for badminton at 6pm

$q'$: find all cities that are suitable for badminton at 6pm for all possible weather for the same city

**Definition**

$q'$ is a first-order (FO) rewriting of $q$ if

$q'(db) = \text{Consistent Answer of } q \text{ over } db = \bigcap_{\text{rep is a repair of } db} q(\text{rep})$

Not all $q$ has an FO-rewriting...
Finding Consistent Answers from Inconsistent Data: Systems, Algorithms, and Complexity

For which queries can we find the consistent answers efficiently?

How efficient can we find the consistent answers?

Can we build a system finding the consistent answers?
Finding Consistent Answers from Inconsistent Data: Systems, Algorithms, and Complexity

For which queries can we find the consistent answers efficiently?

How efficient can we find the consistent answers?

Can we build a system finding the consistent answers?
Finding Consistent Answers from Inconsistent Data: Systems, Algorithms, and Complexity

For which queries can we find the consistent answers efficiently?

How efficient can we find the consistent answers?

Can we build a system finding the consistent answers?
Finding Consistent Answers from Inconsistent Data: Systems, Algorithms, and Complexity

For which queries can we find the consistent answers efficiently?

How efficient can we find the consistent answers?

Can we build a system finding the consistent answers?
System motivations

$q$
System motivations

\[ q \] Is the query \( q \) FO-rewritable? \[ q' \] yes no

Consis. answer be computed in PTIME? \[ yes \] \[ no \]

PTIME algorithm\[ yes \] enum. algorithm \[ no \]
System motivations

$q$ Is the query $q$ FO-rewritable? $q'$

Consistency answer be computed in PTIME?

PTIME algorithm

Enum. algorithm
System motivations

Is the query $q$ FO-rewritable? 
- yes 
- no

Consis. answer be computed in PTIME? 
- yes 
- no

PTIME algorithm
- yes
- enum. algorithm
Theoretical motivations

Problem: CERTAINTY(q), for a fixed query q as an FO sentence (T/F)

Input: a database db (as finite relations)

Question: does rep |= q hold for every rep of db?

Repair (rep): a maximal subset of db that satisfies the PK constraint
Theoretical motivations

Problem: CERTAINTY($q$), for a fixed query $q$ as an FO sentence ($T/F$)

Input: a database $db$ (as finite relations)

Question: does $rep \models q$ hold for every $rep$ of $db$?

Repair ($rep$): a maximal subset of $db$ that satisfies the PK constraint

Proposition

For every fixed query $q$, CERTAINTY($q$) is in coNP.

Proof: Guess a $rep$ of $db$ and check if $rep \models q$ in P (even in $AC^0$) since $q$ is fixed.
Theoretical motivations

Assuming $P \neq NP \ldots$

$P \subset \text{coNP}$
Theoretical motivations

Assuming $P \neq NP$...
Theoretical motivations

Assuming $P \neq NP$. . .

Possibly $NP$-intermediate: Graph Isomorphism, Factoring
Theoretical motivations

Assuming $P \neq NP$. . .

Possibly $NP$-intermediate: Graph Isomorphism, Factoring

**Conjecture**

*For every union of BCQ $q$, CERTAINTY($q$) is in $P$ or $coNP$-complete.*

unions of BCQ: $q_1 \lor \cdots \lor q_n$ for BCQs $q_i$ in $FO(\exists, \land)$
Relationship with Constraint Satisfaction Problems (CSP)

Conjecture

For every union of BCQ $q$, $\text{CERTAINTY}(q)$ is in $P$ or $\text{coNP}$-complete.

- Conservative CSP $\leq_p \text{CERTAINTY}(q)$ [Fontaine'15]
- CSP $\leq_p \text{CQA}$ for UCQs w.r.t. GAV constraints [Fontaine'15]

- Conservative CSP is in $P$ or NP-complete. [Bulatov'03]
- CSP is in $P$ or NP-complete. [Bulatov'17 & Zhuk'17]
Relationship with Constraint Satisfaction Problems (CSP)

**Conjecture**

*For every union of BCQ $q$, CERTAINTY($q$) is in $P$ or coNP-complete.*

- Conservative CSP $\leq_p$ CERTAINTY($q$)  
  [Fontaine'15]

- CSP $\leq_p$ CQA for UCQs w.r.t. GAV constraints  
  [Fontaine'15]

- Conservative CSP is in $P$ or NP-complete.  
  [Bulatov'03]

- CSP is in $P$ or NP-complete.  
  [Bulatov'17 & Zhuk'17]
Our focus

Conjecture

For every union of BCQ q, CERTAINTY(q) is in P or coNP-complete.

Settled when q is self-join-free (SJF)! [Koutris & Wijsen, PODS’15, ICDT’19]

\[ q(x) = \exists y, z : \text{Forecast}(x, y) \land \text{Activity}(y, z, "Yes") \]
\[ q' = \exists y : \text{Flight}("Madison", y) \land \text{Flight}(y, "LA") \]

✓

×
<table>
<thead>
<tr>
<th>( C_{\text{forest}} )</th>
<th>( \alpha )-acyclic</th>
<th>SJF two tables</th>
<th>SJF simple keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>FO</td>
<td>FO, non-FO</td>
<td>P, coNP-complete</td>
<td>P, coNP-complete</td>
</tr>
<tr>
<td>[FM, ICDT’05]</td>
<td>[Wijsen, PODS’10]</td>
<td>[KP, IPL’12]</td>
<td>[KS, ICDT’14]</td>
</tr>
</tbody>
</table>

theory
$C_{forest}$

$\alpha$-acyclic

SJF two tables

SJF simple keys

$C_{forest}$ via FO

theory

system

$C_{forest}$ via FO

$\alpha$-acyclic FO, non-FO

SJF two tables P, coNP-complete

SJF simple keys P, coNP-complete

ConQuer

FO

FO, non-FO

P, coNP-complete

P, coNP-complete

$C_{forest}$ via FO

[FM, ICDT’05]

[Wijsen, PODS’10]

[KP, IPL’12]

[KS, ICDT’14]

[FM, SIGMOD’05]
SJF

$\alpha$-acyclic

SJF two tables

SJF simple keys

$C_{\text{forest}}$ via $\text{FO}$

$C_{\text{forest}}$ via $\text{FO}$

theory

system
SJF paths

FO, NL-complete, P-complete, coNP-complete

[KOW, PODS'21]

SJF

FO, L-complete, coNP-complete

[KW, ICDT'19]

SJF

FO, P\ FO, coNP-complete

[KW, PODS'15]

$C_{forest}$

FO, \(\alpha\)-acyclic

[FM, ICDT'05]

$\alpha$-acyclic

FO, non-FO

[Wijsen, PODS'10]

SJF two tables

P, coNP-complete

[KP, IPL'12]

SJF simple keys

P, coNP-complete

[KS, ICDT'14]

ConQuer

$C_{forest}$ via FO

[FM, SIGMOD'05]

theory

system
<table>
<thead>
<tr>
<th>SJF rooted trees (and beyond)</th>
<th>SJF paths</th>
</tr>
</thead>
<tbody>
<tr>
<td>FO, P \ FO, coNP-complete</td>
<td>FO, NL-complete, P-complete, coNP-complete</td>
</tr>
<tr>
<td>[KOW, PODS'24]</td>
<td>[KOW, PODS'21]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SJF</th>
</tr>
</thead>
<tbody>
<tr>
<td>FO, L-complete, coNP-complete</td>
</tr>
<tr>
<td>[KW, ICDT'19]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FO, P \ FO, coNP-complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>[KW, PODS'15]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( C_{\text{forest}} )</th>
<th>( \alpha )-acyclic</th>
<th>SJF two tables</th>
<th>SJF simple keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>FO</td>
<td>FO, non-FO</td>
<td>P, coNP-complete</td>
<td>P, coNP-complete</td>
</tr>
<tr>
<td>[FM, ICDT'05]</td>
<td>[Wijsen, PODS'10]</td>
<td>[KP, IPL'12]</td>
<td>[KS, ICDT'14]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ConQuer</th>
</tr>
</thead>
<tbody>
<tr>
<td>( C_{\text{forest}} ) via FO</td>
</tr>
<tr>
<td>[FM, SIGMOD'05]</td>
</tr>
</tbody>
</table>

**Theory**

<table>
<thead>
<tr>
<th>system</th>
</tr>
</thead>
</table>

Xiating Ouyang

Consistent Query Answering

PhD Defense
SJF rooted trees (and beyond)
- $\mathbf{FO}$, $\mathbf{P} \setminus \mathbf{FO}$, coNP-complete
- [KOW, PODS'24]

SJF paths
- $\mathbf{FO}$, NL-complete, $\mathbf{P}$-complete, coNP-complete
- [KOW, PODS'21]

SJF
- $\mathbf{FO}$, L-complete, coNP-complete
- [KW, ICDT'19]

Conquesto
- SJF via $\mathbf{FO}$
- [AJLSW, CIKM'20]

$C_{\text{forest}}$
- $\alpha$-acyclic
- $\mathbf{FO}$, non-$\mathbf{FO}$
- $\mathbf{P}$, coNP-complete
- [FM, ICDT'05], [Wijsen, PODS'10], [KP, IPL'12], [KS, ICDT'14]

theory

system

Xiating Ouyang
Consistent Query Answering
PhD Defense
<table>
<thead>
<tr>
<th>SJF rooted trees (and beyond)</th>
<th>ConQuer [FM, ICDT'05]</th>
</tr>
</thead>
<tbody>
<tr>
<td>( FO, \ \mathbb{P} \setminus \ FO, \ \text{coNP-complete} )</td>
<td>( C_{\text{forest}} ) via ( \text{FO} ) [FM, SIGMOD'05]</td>
</tr>
<tr>
<td>( \alpha )-acyclic ( \text{FO, non-FO} ) [Wijsen, PODS'10]</td>
<td>( \text{SJF two tables} ) ( P, \ \text{coNP-complete} ) [KP, IPL'12]</td>
</tr>
<tr>
<td>( \text{SJF paths} )</td>
<td>( \text{SJF simple keys} ) ( P, \ \text{coNP-complete} ) [KS, ICDT'14]</td>
</tr>
<tr>
<td>( FO, \ \text{NL-complete}, \ \mathbb{P}\text{-complete, coNP-complete} )</td>
<td>( \text{Conquesto} ) ( \text{SJF via FO} ) [AJLSW, CIKM'20]</td>
</tr>
<tr>
<td>[KOW, PODS'21]</td>
<td>[KW, PODS'24]</td>
</tr>
</tbody>
</table>

| SJF \( \mathbb{P} \setminus \mathbb{FO} \), \ \text{coNP-complete} \] | \[KW, ICDT'19\]  |
|SJF \( \mathbb{P} \setminus \mathbb{FO} \), \ \text{coNP-complete} \] | \[KW, PODS'15\]  |
|SJF \( \mathbb{P} \setminus \mathbb{FO} \), \ \text{coNP-complete} \] | \[KW, ICDT'19\]  |

| \( \text{CAvSAT} \) | \[DK, SAT'19, ICDE'21\]  |
| \( \text{EQUIP} \) | \[KPT, VLDB'13\]  |

| \( \text{Conquesto} \) | \( \text{SJF via FO} \) \[AJLSW, CIKM'20\]  |
| \( \text{CAvSAT} \) | \( \text{* via SAT} \) \[DK, SAT'19, ICDE'21\]  |
| \( \text{EQUIP} \) | \( \text{* via BIP} \) \[KPT, VLDB'13\]  |

| \( \text{C}_{\text{forest}} \) via \( \text{FO} \) \[FM, SIGMOD'05\]  |
| \( \text{CAvSAT} \) | \( \text{Conquesto} \) \[AJLSW, CIKM'20\]  |

| \( \text{theory} \)  |
| \( \text{system} \)  |
SJF rooted trees (and beyond)
- FO, P \ FO, coNP-complete
  - [KOW, PODS’24]

SJF paths
- FO, NL-complete, P-complete, coNP-complete
  - [KOW, PODS’21]

SJF
- FO, L-complete, coNP-complete
  - [KW, ICDT’19]

SJF
- FO, P \ FO, coNP-complete
  - [KW, PODS’15]

C_{forest}
- \alpha\text{-acyclic}
  - [FM, ICDT’05]
- FO, non-FO
  - [Wijsen, PODS’10]

SJF two tables
- P, coNP-complete
  - [KP, IPL’12]

SJF simple keys
- P, coNP-complete
  - [KS, ICDT’14]

ConQuer
- C_{forest} via FO
  - [FM, SIGMOD’05]

CAvSAT
- * via SAT
  - [DK, SAT’19, ICDE’21]

EQUIP
- * via BIP
  - [KPT, VLDB’13]

LinCQA
- PPJT via FO in O(N)
  - [FKOW, SIGMOD’23]

Conquesto
- SJF via FO
  - [AJLSW, CIKM’20]

theory

system

Xiating Ouyang
Consistent Query Answering
PhD Defense
It starts from *Acyclic Queries*...
Acyclic query evaluation

SELECT DISTINCT 1
FROM Forecast, Activity
WHERE Forecast.weather = Activity.weather
    AND Activity.Badmin = "Yes"

\[ q = \exists x, y, z : \text{Forecast}(x, y) \land \text{Activity}(y, z, "Yes") \]
Acyclic query evaluation

```
SELECT DISTINCT 1
FROM Forecast, Activity
WHERE Forecast.weather = Activity.weather
    AND Activity.Badmin = "Yes"
```

\[ q = \exists x, y, z : \text{Forecast}(x, y) \land \text{Activity}(y, z, "Yes") \]

Join Tree of \( q \)

\[ \text{Forecast}(x, y) \quad \text{Activity}(y, z, "Yes") \]
Acyclic query evaluation

```
SELECT DISTINCT 1
FROM Forecast, Activity
WHERE Forecast.weather = Activity.weather
AND Activity.Badmin = "Yes"
```

\[ q = \exists x, y, z : \text{Forecast}(x, y) \land \text{Activity}(y, z, "Yes") \]

Join Tree of \( q \)
Yannakakis [VLDB’81]

The answer to every **Boolean** acyclic query can be computed in $O(|db|)$. 
consistent answer

The answer to every Boolean acyclic query can be computed in $O(|db|)$. with a pair-pruning join tree (PPJT)
Yannakakis [VLDB’81]  Our result

consistent answer

The answer to every **Boolean** acyclic query can be computed in \( O(|db|) \).

\[ \land \]

with a pair-pruning join tree (PPJT)

non-Boolean \( \leq^P_T \) **Boolean**
SELECT
  DISTINCT Posts.Id, Posts.Title
FROM
  Posts, PostHistory, Votes, Comments
WHERE
  Posts.Tags LIKE "%SQL%"
  AND Posts.id = PostHistory.PostId
  AND Posts.id = Comments.PostId
  AND Posts.id = Votes.PostId
  AND Votes.BountyAmount > 100
  AND PostHistory.PostHistoryTypeId = 2
  AND Comments.score = 0
Xiating Ouyang  
Consistent Query Answering  
PhD Defense  
25 / 71
Original query (prev. slide) + primary key info $\xrightarrow{\text{LinCQA}}$ Query rewriting
PPJT is a wide class

+ $\subseteq$ Selection, Projection, Join queries
+ star/snowflake schema (e.g. 14/21 TPC-H)
+ Every acyclic query in $C_{\text{forest}}$ [Fuxman & Miller’05] has a PPJT

- no self-joins...
- no aggregation (yet) [Dixit & Kolaitis, 2022] [El Khalfioui & Wijsen, 2022]
PPJT is a wide class

+ ⊂ Selection, Projection, Join queries
+ star/snowflake schema (e.g. 14/21 TPC-H)
+ Every acyclic query in $C_{forest}$ [Fuxman & Miller’05] has a PPJT

- no self-joins...
- no aggregation (yet) [Dixit & Kolaitis, 2022] [El Khalfioui & Wijsen, 2022]
From *Join Tree* to Pair-pruning Join Tree (PPJT)
A join tree **rooted** at some atom is a PPJT if

the root of every subtree is **unattacked** in the subtree
Pair-pruning join tree (PPJT)

A join tree rooted at some atom is a PPJT if

the root of every subtree is unattacked in the subtree

Diagram:

```
Forecast(x, y)
  ↓
Activity(y, z, "Yes")
```
A join tree rooted at some atom is a PPJT if

- the root of every subtree is unattacked in the subtree.

Diagram:

```
        Forecast(x, y)
           /
          /
Activity(y, z, "Yes")
```
A join tree rooted at some atom is a PPJT if

the root of every subtree is unattacked in the subtree
LinCQA: From PPJT to FO-rewriting

Remove a primary key if some tuple with this primary key is “bad”

Also expressible in SQL! Runs in $O(N)$
LinCQA: From PPJT to FO-rewriting

Remove a primary key if some tuple with this primary key is “bad”

\[
\text{Forecast}(x, y) \quad \text{Activity}(y, z, "Yes")
\]

\[
\begin{align*}
\text{Activity} & : \text{Root} \\
\text{Activity} & : \text{Child} \\
\text{Forecast} & : \text{Root} \\
\text{Forecast} & : \text{Child}
\end{align*}
\]

also expressible in SQL!
runs in $O(N)$
LinCQA: From PPJT to FO-rewriting

Remove a primary key if some tuple with this primary key is “bad”

Forecast($x, y$) –> Activity($y, z, "Yes"]

also expressible in SQL! runs in $O(N)$
LinCQA: From PPJT to FO-rewriting

Remove a primary key if some tuple with this primary key is “bad”

\[
\text{Activity}(y, z, "Yes")
\]

\[
\text{Forecast}(x, y)
\]

also expressible in SQL! runs in \(O(N)\)
LinCQA: From PPJT to FO-rewriting

Remove a primary key if some tuple with this primary key is “bad”

\[
\text{Forecast}(x, y) \quad \Downarrow \quad \text{Activity}(y, z, "Yes")
\]

also expressible in SQL! runs in \(O(N)\)
LinCQA: From PPJT to FO-rewriting

Remove a primary key if some tuple with this primary key is “bad”

\[
\text{Activity}(y, z, "Yes") \quad \rightarrow \quad \text{Forecast}(x, y)
\]

also expressible in SQL!
runs in $O(N)$
Remove a primary key if some tuple with this primary key is "bad"

Forecast($x, y$)

Activity($y, z, "Yes"$)

also expressible in SQL!

runs in $O(N)$
LinCQA: From PPJT to FO-rewriting

Remove a primary key if some tuple with this primary key is “bad”

\[
\text{Forecast}(\underline{x}, y) \\
\text{Activity}(\underline{y}, z, "Yes")
\]

\[
\text{Forecast}_{\text{join}}() \leftarrow \text{Forecast}(x, y), \neg \text{Forecast}_{\text{fkey}}(x)
\]

\[
\text{Forecast}_{\text{fkey}}(x) \leftarrow \text{Forecast}(x, y), \neg \text{Activity}_{\text{join}}(y)
\]

\[
\forall \text{Child} : \text{Root}_{\text{fkey}}(\underline{x}) \leftarrow \text{Root}(\underline{x}, \underline{y}), \neg \text{Child}_{\text{join}}(\underline{\alpha})
\]

\[
\text{Child}_{\text{join}}(\underline{\alpha}) \leftarrow \text{Child}(\underline{u}, \underline{v}), \neg \text{Child}_{\text{fkey}}(\underline{u})
\]

\[
\text{Activity}_{\text{join}}(y) \leftarrow \text{Activity}(y, z, w), \neg \text{Activity}_{\text{fkey}}(y)
\]

\[
\text{Activity}_{\text{fkey}}(y) \leftarrow \text{Activity}(y, z, w), w \neq "Yes"
\]

also expressible in SQL! runs in \(O(N)\)
LinCQA: From PPJT to FO-rewriting

Remove a primary key if some tuple with this primary key is "bad"

\[ \text{Forecast}(x, y) \]

\[ \text{Activity}(y, z, "Yes") \]

also expressible in SQL! runs in \( O(N) \)
LinCQA: From PPJT to FO-rewriting

Remove a primary key if some tuple with this primary key is “bad”

\[
\text{Forecast}(x, y) \quad \text{Activity}(y, z, "Yes")
\]

\[
\text{Forecast}_{\text{fkey}}(x) \quad \text{Activity}_{\text{join}}(y)
\]

\[
\forall \text{Child} : \text{Root}_{\text{fkey}}(x) \quad \text{Child}_{\text{join}}(\vec{\alpha})
\]

\[
\text{Forecast}_{\text{join}}() \quad \text{Activity}_{\text{join}}(y)
\]

also expressible in SQL! runs in \(O(N)\)
LinCQA: From PPJT to FO-rewriting

Remove a primary key if some tuple with this primary key is “bad”

\[
\text{Forecast}(x, y) \\
\text{Activity}(y, z, "Yes")
\]

\[
\text{Forecast}_\text{fkey}(x) :- \text{Forecast}(x, y), \neg \text{Activity}_\text{join}(y)
\]

\[
\forall \text{Child} : \text{Root}_\text{fkey}(\vec{x}) :- \text{Root}(\vec{x}, \vec{y}), \neg \text{Child}_\text{join}(\vec{\alpha})
\]

\[
\text{Child}_\text{join}(\vec{\alpha}) :- \text{Child}(\vec{u}, \vec{v}), \neg \text{Child}_\text{fkey}(\vec{u})
\]

\[
\text{Activity}_\text{join}(y) :- \text{Activity}(y, z, w), \neg \text{Activity}_\text{fkey}(y)
\]

\[
\text{Activity}_\text{fkey}(y) :- \text{Activity}(y, z, w), w \neq "Yes"
\]

also expressible in SQL!

runs in \(O(N)\)
LinCQA: From PPJT to FO-rewriting

Remove a primary key if some tuple with this primary key is "bad"

Forecast\((x, y)\)

\rightarrow

Activity\((y, z, "Yes")\)

Forecast\_join() \rightarrow Forecast(x, y), \neg Forecast\_fkey(x)

Forecast\_fkey(x) \rightarrow Forecast(x, y), \neg Activity\_join(y)

\forall Child : Root\_fkey(\vec{x}) \rightarrow Root(\vec{x}, \vec{y}), \neg Child\_join(\vec{\alpha})

Child\_join(\vec{\alpha}) \rightarrow Child(\vec{u}, \vec{v}), \neg Child\_fkey(\vec{u})

Activity\_join(y) \rightarrow Activity(y, z, w), \neg Activity\_fkey(y)

Activity\_fkey(y) \rightarrow Activity(y, z, w), w \neq "Yes"

also expressible in SQL!
runs in $O(N)$
From Boolean to non-Boolean

```
SELECT DISTINCT A1, A2 FROM T WHERE A3 = 42
```

Step 1 Evaluate directly

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>x</td>
<td>y</td>
</tr>
</tbody>
</table>

Step 2 Reduce to **Boolean** (using PPJT)

```
SELECT DISTINCT 1 FROM T WHERE A3 = 42 AND A1 = a AND A2 = b
```

if **yes**, then output \((a, b)\), otherwise continue

```
SELECT DISTINCT 1 FROM T WHERE A3 = 42 AND A1 = x AND A2 = y
```

... 

LinCQA $\rightarrow$ a single SQL/Datalog query
<table>
<thead>
<tr>
<th>Acyclic $q$</th>
<th>PPJT</th>
<th>Yannakakis [VLDB’81]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean $q$</td>
<td>$O(N)$</td>
<td>$O(N)$</td>
</tr>
<tr>
<td>non-Boolean $q$</td>
<td>$O(N \cdot</td>
<td>\text{OUT}_{\text{inconsistent}}</td>
</tr>
<tr>
<td>free-connex $q$</td>
<td>$O(N +</td>
<td>\text{OUT}_{\text{consistent}}</td>
</tr>
</tbody>
</table>

Consistent answers of common join queries can be computed with no asymptotic overhead.
<table>
<thead>
<tr>
<th>Acyclic $q$</th>
<th>PPJT</th>
<th>Yannakakis [VLDB'81]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean $q$</td>
<td>$O(N)$</td>
<td>$O(N)$</td>
</tr>
<tr>
<td>non-Boolean $q$</td>
<td>$O(N \cdot</td>
<td>\text{OUT}_{\text{inconsistent}}</td>
</tr>
<tr>
<td>free-connex $q$</td>
<td>$O(N +</td>
<td>\text{OUT}_{\text{consistent}}</td>
</tr>
</tbody>
</table>

Consistent answers of common join queries can be computed with no asymptotic overhead.
Experiments
### Setup & Baselines

<table>
<thead>
<tr>
<th>System</th>
<th>Target class</th>
<th>Interim. output</th>
<th>Backend</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAvSAT</td>
<td>*</td>
<td>SAT formula</td>
<td>SQL Server &amp; MaxHS</td>
</tr>
<tr>
<td>Conquer</td>
<td>$C_{\text{forest}}$</td>
<td>SQL</td>
<td>SQL Server</td>
</tr>
<tr>
<td>Improved Conquesto</td>
<td>SJF FO</td>
<td>SQL</td>
<td>SQL Server</td>
</tr>
<tr>
<td>LinCQA</td>
<td>PPJT</td>
<td>SQL</td>
<td>SQL Server</td>
</tr>
</tbody>
</table>

CloudLab
Stackoverflow data

- Metadata of stackoverflow.com as of 02/2021 from Stack Exchange Data Dump
- 551M rows, 400 GB

<table>
<thead>
<tr>
<th>Table</th>
<th># of rows</th>
<th>inconsistencyRatio</th>
<th>blockSize</th>
<th># of Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Users</td>
<td>14M</td>
<td>0%</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>Posts</td>
<td>53M</td>
<td>0%</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>PostHistory</td>
<td>141M</td>
<td>0.001%</td>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>Badges</td>
<td>40M</td>
<td>0.58%</td>
<td>941</td>
<td>4</td>
</tr>
<tr>
<td>Votes</td>
<td>213M</td>
<td>30.9%</td>
<td>1441</td>
<td>6</td>
</tr>
</tbody>
</table>
Stackoverflow results

\[ Q_1 : Posts \bowtie \text{Votes} \quad Q_2 : \text{Users} \bowtie \text{Badges} \quad Q_3 : \text{Users} \bowtie \text{Posts} \]
\[ Q_4 : \text{Users} \bowtie \text{Posts} \bowtie \text{Comments} \quad Q_5 : \text{Posts} \bowtie \text{PostHistory} \bowtie \text{Votes} \bowtie \text{Comments} \]

| \# poss. | 27578 | 145 | 38320 | 3925 | 1250 |
| \# cons. | 27578 | 145 | 38320 | 3925 | 1245 |
Stackoverflow results

$Q_1 : \text{Posts} \bowtie \text{Votes}$  \quad $Q_2 : \text{Users} \bowtie \text{Badges}$  \quad $Q_3 : \text{Users} \bowtie \text{Posts}$

$Q_4 : \text{Users} \bowtie \text{Posts} \bowtie \text{Comments}$

$Q_5 : \text{Posts} \bowtie \text{PostHistory} \bowtie \text{Votes} \bowtie \text{Comments}$

![Graph showing query results](graph.png)

<table>
<thead>
<tr>
<th></th>
<th># poss.</th>
<th># cons.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>27578</td>
<td>27578</td>
</tr>
<tr>
<td>Q2</td>
<td>145</td>
<td>145</td>
</tr>
<tr>
<td>Q3</td>
<td>38320</td>
<td>38320</td>
</tr>
<tr>
<td>Q4</td>
<td>3925</td>
<td>3925</td>
</tr>
<tr>
<td>Q5</td>
<td>1250</td>
<td>1245</td>
</tr>
</tbody>
</table>

- Original Query: LinCQA
- Conquer
- FastFO
- CAvSAT

Time Out: N/A
## Concluding remarks

<table>
<thead>
<tr>
<th>Acyclic $q$</th>
<th>LinCQA [FKOW’23]</th>
<th>Yannakakis [VLDB’81]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean $q$</td>
<td>$O(N)$</td>
<td>$O(N)$</td>
</tr>
<tr>
<td>non-Boolean $q$</td>
<td>$O(N \cdot</td>
<td>\text{OUT}_{\text{inconsistent}}</td>
</tr>
<tr>
<td>free-connex $q$</td>
<td>$O(N +</td>
<td>\text{OUT}_{\text{consistent}}</td>
</tr>
</tbody>
</table>

### Original Query

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>Q5</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>101</td>
<td>102</td>
<td>103</td>
<td>N/A</td>
</tr>
</tbody>
</table>

### Performance

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>Q5</th>
</tr>
</thead>
<tbody>
<tr>
<td>27578</td>
<td>145</td>
<td>38320</td>
<td>3925</td>
<td>1250</td>
</tr>
</tbody>
</table>

### Time Out

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>Q5</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/A</td>
<td>Time Out</td>
<td>N/A</td>
<td>Time Out</td>
<td>N/A</td>
</tr>
</tbody>
</table>

### N/A

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>Q5</th>
</tr>
</thead>
<tbody>
<tr>
<td># poss.</td>
<td>27578</td>
<td>145</td>
<td>38320</td>
<td>3925</td>
</tr>
<tr>
<td># cons.</td>
<td>27578</td>
<td>145</td>
<td>38320</td>
<td>3925</td>
</tr>
</tbody>
</table>

### Graph

- Original Query
- LinCQA
- Conquer
- FastFO
- CAvSAT

Xiating Ouyang

Consistent Query Answering

PhD Defense
SJF rooted trees (and beyond)
- $\mathit{FO}$, $\mathit{P} \setminus \mathit{FO}$, coNP-complete
  - [KOW, PODS'24]

SJF paths
- $\mathit{FO}$, NL-complete, $\mathit{P}$-complete, coNP-complete
  - [KOW, PODS'21]

SJF
- $\mathit{FO}$, L-complete, coNP-complete
  - [KW, ICDT'19]

SJF
- $\mathit{FO}$, $\mathit{P} \setminus \mathit{FO}$, coNP-complete
  - [KW, PODS'15]

$C_{\text{forest}}$
- $\alpha$-acyclic
- $\mathit{FO}$, non-FO
  - [FM, ICDT'05] [Wijsen, PODS'10]

SJF two tables
- $\mathit{P}$, coNP-complete
  - [KP, IPL'12]

SJF simple keys
- $\mathit{P}$, coNP-complete
  - [KS, ICDT'14]

$\mathit{C}_{\text{forest}}$ via $\mathit{FO}$
- [FM, SIGMOD'05]

theory

system

CAvSAT
- * via SAT
  - [DK, SAT'19, ICDE'21]

EQUIP
- * via BIP
  - [KPT, VLDB'13]

LinCQA
- PPJT via $\mathit{FO}$ in $O(N)$
  - [FKOW, SIGMOD'23]

Conquesto
- SJF via $\mathit{FO}$
  - [AJLSW, CIKM'20]

ConQuer
- $\mathit{C}_{\text{forest}}$ via $\mathit{FO}$
  - [FM, SIGMOD'05]
Why are self-joins complicated?
Problem: CERTAINTY(\(q\)), where

\[
q = \exists x, y, z : \text{Forecast}(x, y) \land \text{Activity}(y, z, "Yes")
\]

Input: a database \(db\) (as a finite set of relations)

Question: does \(rep \models q\) hold for every \(rep\) of \(db\) ?

<table>
<thead>
<tr>
<th>Forecast</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>City</strong></td>
<td><strong>Weather</strong></td>
</tr>
<tr>
<td>* MSN</td>
<td>Rainy</td>
</tr>
<tr>
<td>* MSN</td>
<td>Sunny</td>
</tr>
<tr>
<td>LA</td>
<td>Sunny</td>
</tr>
<tr>
<td>Seattle</td>
<td>Rainy</td>
</tr>
</tbody>
</table>

Forecast(MSN, Rainy) could only satisfy the predicate \(\text{Forecast}(x, y)\).
Problem: CERTAINTY($q$), where

\[ q = \exists x, y, z : \text{Forecast}(x, y) \land \text{Activity}(y, z, "Yes") \]

Input: a database $db$ (as a finite set of relations)

Question: does $\text{rep} \models q$ hold for every $\text{rep}$ of $db$?

<table>
<thead>
<tr>
<th>Forecast</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>City</strong></td>
<td><strong>Weather</strong></td>
</tr>
<tr>
<td>* MSN</td>
<td>Rainy</td>
</tr>
<tr>
<td>* MSN</td>
<td>Sunny</td>
</tr>
<tr>
<td>LA</td>
<td>Sunny</td>
</tr>
<tr>
<td>Seattle</td>
<td>Rainy</td>
</tr>
</tbody>
</table>

Forecast(MSN, Rainy) could only satisfy the predicate Forecast($x$, $y$)
Problem: CERTAINTY($q$), where

$$q = \exists x, y, z : R(x, y) \land R(y, z) \land X(z, w) = RRX$$

Input: a database $db$ (as a finite set of relations)

Question: does $rep \models q$ hold for every $rep$ of $db$?

$R(1, 2)$ can satisfy either $R(x, y)$ or $R(y, z)$ now

rep\textsubscript{1}

rep\textsubscript{2}
Problem: CERTAINTY($q$), where

$$q = \exists x, y, z : R(x, y) \land R(y, z) \land X(z, w) = RRX$$

Input: a database $db$ (as a finite set of relations)

Question: does $rep \models q$ hold for every $rep$ of $db$?

$R$

<table>
<thead>
<tr>
<th></th>
<th>$A_1$</th>
<th>$A_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

$X$

<table>
<thead>
<tr>
<th></th>
<th>$B_1$</th>
<th>$B_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

$R(1, 2)$ can satisfy either $R(x, y)$ or $R(y, z)$ now

$rep_1$

$rep_2$
Problem: CERTAINTY($q$), where

\[ q = \exists x, y, z : R(x, y) \land R(y, z) \land X(z, w) = RRX \]

Input: a database $db$ (as a finite set of relations)

Question: does $\text{rep} \models q$ hold for every $\text{rep}$ of $db$?

\[
\begin{array}{c|cc}
R & A_1 & A_2 \\
\hline
0 & 1 & \\
1 & 2 & \\
1 & 3 & \\
2 & 3 & \\
\end{array}
\quad
\begin{array}{c|cc}
X & B_1 & B_2 \\
\hline
\end{array}
\quad
\begin{array}{c|c|c|c|c}
R & R & R & R & R \\
0 & 1 & 3 & X & 4 \\
\end{array}
\]

$R(1, 2)$ can satisfy either $R(x, y)$ or $R(y, z)$ now.

\[
\begin{array}{c|c|c|c}
\text{rep}_1 & \text{rep}_2 \\
0 \rightarrow R \rightarrow 1 \rightarrow R \rightarrow 3 \rightarrow X \rightarrow 4 & 0 \rightarrow R \rightarrow 1 \rightarrow R \rightarrow 3 \rightarrow X \rightarrow 4 \\
\end{array}
\]

RRX \quad RRX \quad RRX \quad RRX \quad RRX \quad RRX \quad RRX
Problem: CERTAINTY($q$), where
\[ q = \exists x, y, z : R(x, y) \land R(y, z) \land X(z, w) = RRX \]

Input: a database $db$ (as a finite set of relations)

Question: does $\text{rep} \models q$ hold for every $\text{rep}$ of $db$?

\[
R \begin{array}{cc}
A_1 & A_2 \\
0 & 1 \\
1 & 2 \\
1 & 3 \\
2 & 3 \\
\end{array}
\]

\[
X \begin{array}{cc}
B_1 & B_2 \\
2 & 3 \\
1 & 4 \\
\end{array}
\]

$R(1, 2)$ can satisfy either $R(x, y)$ or $R(y, z)$ now

$\text{rep}_1$

$\text{rep}_2$
Problem: CERTAINTY($q$), where

\[ q = \exists x, y, z : R(x, y) \land R(y, z) \land X(z, w) = RRX \]

Input: a database $db$ (as a finite set of relations)

Question: does $\text{rep} \models q$ hold for every $\text{rep}$ of $db$?

\[ R \begin{array}{c|c}
A_1 & A_2 \\
0 & 1 \\
1 & 2 \\
1 & 3 \\
2 & 3 \\
\end{array} \]

\[ X \begin{array}{c|c}
B_1 & B_2 \\
3 & 4 \\
\end{array} \]

$R(1, 2)$ can satisfy either $R(x, y)$ or $R(y, z)$ now

\[ \begin{array}{ccc}
R & 1 & X \\
0 & 1 & 3 \\
& 2 & \\
\hline
& & \\
\end{array} \]

\[ \begin{array}{ccc}
RRX & RRX & RRX \\
\end{array} \]
Problem: CERTAINTY\((q)\), where
\[ q = \exists x, y, z : R(x, y) \land R(y, z) \land X(z, w) = RRX \]

Input: a database \(db\) (as a finite set of relations)
Question: does \(\text{rep} \models q\) hold for every \(\text{rep}\) of \(db\)?

\[
\begin{array}{c|cc}
R & A_1 & A_2 \\
\hline
0 & 1 & \\
1 & 2 & \\
1 & 3 & \\
2 & 3 & \\
\end{array}
\quad
\begin{array}{c|cc}
X & B_1 & B_2 \\
\hline
 & 3 & \\
 & 4 & \\
\end{array}
\]

\(R(1, 2)\) can satisfy either \(R(x, y)\) or \(R(y, z)\) now

\(\text{rep}_1\)

\(\text{rep}_2\)

\(RRX\) \quad \(RRX\) \quad \(RRX\)
Problem: CERTAINTY($q$), where

$$q = \exists x, y, z : R(x, y) \land R(y, z) \land X(z, w) = RRX$$

Input: a database $db$ (as a finite set of relations)
Question: does $rep \models q$ hold for every $rep$ of $db$?

$$R \begin{array}{c|cc}
  A_1 & A_2 \\
\hline
  0 & 1 \\
  1 & 2 \\
  1 & 3 \\
  2 & 3 \\
\end{array}$$

$$X \begin{array}{c|cc}
  B_1 & B_2 \\
\hline
  2 & 3 \\
  3 & 4 \\
\end{array}$$

$R(1, 2)$ can satisfy either $R(x, y)$ or $R(y, z)$ now

$R(1, 2)$ can satisfy either $R(x, y)$ or $R(y, z)$ now

$rep_1$

$$R \begin{array}{c|cc}
  & & \\
\hline
  0 & R & 1 \\
  1 & R & 3 \\
  2 & R & X \\
\end{array}$$

$RRX$ $RRX$ $RRX$

$rep_2$

$$R \begin{array}{c|cc}
  & & \\
\hline
  0 & R & 1 \\
  1 & R & 3 \\
  2 & R & X \\
\end{array}$$

$RRX$ $RRX$ $RRX$ $RRX$ $RRX$ $RRX$ $RRX$ $RRX$ $RRX$
Problem: CERTAINTY(q), where

\[ q = \exists x, y, z : R(x, y) \land R(y, z) \land X(z, w) = RRX \]

Input: a database \textbf{db} (as a finite set of relations)

Question: does \textbf{rep} \models q hold for every \textbf{rep} of \textbf{db}?

\begin{align*}
R & \begin{array}{c|c}
A_1 & A_2 \\
0 & 1 \\
1 & 2 \\
1 & 3 \\
2 & 3 \\
\end{array} \\
X & \begin{array}{c|c}
B_1 & B_2 \\
 & 3 \\
 & 4 \\
\end{array}
\end{align*}

\begin{equation*}
R(1, 2) \text{ can satisfy either } R(x, y) \text{ or } R(y, z) \text{ now}
\end{equation*}

\textbf{rep}_1

\begin{equation*}
\begin{array}{cccc}
0 & 1 & 3 & 4 \\
R & R & X & \\
\end{array}
\end{equation*}

\textbf{rep}_2

\begin{equation*}
\begin{array}{cccc}
0 & 1 & 3 & 4 \\
R & R & X & \\
\end{array}
\end{equation*}
Problem: CERTAINTY\((q)\), where
\[
q = \exists x, y, z : R(x, y) \land R(y, z) \land X(z, w) = RRX
\]

Input: a database \(db\) (as a finite set of relations)
Question: does \(rep \models q\) hold for every \(rep\) of \(db\) ?

\[
\begin{array}{c|cc}
R & A_1 & A_2 \\
\hline
0 & 1 & \\
1 & 2 & \\
1 & 3 & \\
2 & 3 & \\
\end{array}
\]

\[
\begin{array}{c|cc}
X & B_1 & B_2 \\
\hline
2 & 3 & 4 \\
\end{array}
\]

\(R(1, 2)\) can satisfy either \(R(x, y)\) or \(R(y, z)\) now

\(rep_1\)

\(rep_2\)
Problem: CERTAINTY($q$), where

$$q = \exists x, y, z : R(x, y) \land R(y, z) \land X(z, w) = RRX$$

Input: a database $\text{db}$ (as a finite set of relations)

Question: does $\text{rep} \models q$ hold for every $\text{rep}$ of $\text{db}$?

$R(1, 2)$ can satisfy either $R(x, y)$ or $R(y, z)$ now

$\text{rep}_1$

$\text{rep}_2$
Problem: CERTAINTY(q), where

\[ q = \exists x, y, z : R(x, y) \land R(y, z) \land X(z, w) = RRX \]

Input: a database \( db \) (as a finite set of relations)

Question: does \( \text{rep} \models q \) hold for every \( \text{rep} \) of \( db \)?

The key is to exploit this “rewinding” behavior

Proposition

The following statements are equivalent:

1. \( db \) is a “yes”-instance for CERTAINTY(RRX); and
2. \( \exists c \) such that in all repairs, there exists a path of \( RR \cdot R^* \cdot X \) starting at \( c \).
“Reachability”, “NL-complete”
“Reachability”, “\textbf{NL}-complete"  How to find the regular expression?
From path query to NFA

\[ \epsilon \xrightarrow{R} R \xrightarrow{R} RR \xrightarrow{X} RRX \]

NFA(\(RRX\)) accepts \(RRR^*X\)
From path query to NFA

\[
\varepsilon \xrightarrow{R} R \xrightarrow{R} RR \xrightarrow{X} RRX
\]

NFA(\text{RRX}) accepts \text{RRR}^* \text{X}
From path query to NFA

\[ \varepsilon \rightarrow R \rightarrow RR \rightarrow RRR^* X \]

NFA(\text{RRX}) accepts \( RRR^* X \)
From path query to NFA

$NFA(\text{RRX})$ accepts $\text{RRR}^* \text{X}$
From path query to NFA

NFA(\(RRX\)) accepts \(RRR^*X\)
From path query to NFA

NFA(\textit{RRX}) accepts \textit{RRR}^* \textit{X}
From path query to NFA

NFA(\textit{RRX}) accepts \textit{RRR}^* \textit{X}
From path query to NFA (cont.)

NFA($RXRRR$)
Path queries

\[
q = \exists x_0, x_1, \ldots, x_n : R_1(x_0, x_1) \land R_2(x_1, x_2) \land \cdots \land R_n(x_{n-1}, x_n) = R_1 R_2 \ldots R_n
\]

- it can be that \( R_i = R_j \) for \( i \neq j \)
- free variables & constants are easy extensions
Complexity classification for CERTAINTY($q$)

**NL-hard**

\[ q_2 = RX \ Y \]

\[ RXRX\ Y \in NFA(q_2) \]

\[ q_1 = RXRX \]

\[ RXRX(RX)^* = NFA(q_1) \]

**C_1:** $q$ is a prefix of every word in NFA($q$)

**FO-rewritable**
Complexity classification for CERTAINTY($q$)

$\text{coNP}$-complete

$q_4 = RXRX \ YR \ Y \ RXRXRYRXRYRY \in NFA(q_4)$

$C_2$: $q$ is a factor of every word in $NFA(q)$

$P$

$\text{NL}$-hard

$q_2 = RX \ Y \ Y \ RXRX \ Y \ Y \in NFA(q_2)$

$C_1$: $q$ is a prefix of every word in $NFA(q)$

$\text{FO}$-rewritable

$q_1 = RXRX \ RXRX(RX)^* \in NFA(q_1)$
## Complexity classification for CERTAINTY($q$)

<table>
<thead>
<tr>
<th>Complexity</th>
<th>Description</th>
<th>Instance</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>coNP-complete</strong></td>
<td>$q_4 = RXRX \ YRY \ \ RXRXRYRXRYRY \in NFA(q_4)$</td>
<td></td>
</tr>
<tr>
<td><strong>P</strong></td>
<td>$q_3 = RX \ YRY$</td>
<td></td>
</tr>
<tr>
<td><strong>C$_{2.5}$</strong></td>
<td>Whenever $q = uRvRw$, $q$ is a factor of $uRvRvRw$; and whenever $q = uRv_1Rv_2Rw$ for consecutive occurrences of $R$, $v_1 = v_2$ or $Rw$ is a prefix of $Rv_1$.</td>
<td></td>
</tr>
<tr>
<td><strong>NL-hard</strong></td>
<td>$q_2 = RX \ YR \ \ RXRX\ YR \in NFA(q_2)$</td>
<td></td>
</tr>
<tr>
<td><strong>C$_1$</strong></td>
<td>$q$ is a prefix of every word in NFA($q$)</td>
<td></td>
</tr>
<tr>
<td><strong>FO-rewritable</strong></td>
<td>$q_1 = RXRX \ \ RXRX(RX)^* = NFA(q_1)$</td>
<td></td>
</tr>
</tbody>
</table>
Complexity classification for $\text{CERTAINTY}(q)$

- **coNP-complete**
  - $q_4 = RXRX\ YRY\ RXRXRYRXRYRY \in \text{NFA}(q_4)$

- **P-complete**
  - $q_3 = RX\ YRY$

- **NL-complete**
  - $q_2 = RX\ YR\ Y\ RXRX\ YR \in \text{NFA}(q_2)$

- **C_1.5:** Whenever $q = uRvRw$, $q$ is a factor of $uRvRvRw$; and whenever $q = uRv_1v_2Rw$ for consecutive occurrences of $R$, $v_1 = v_2$ or $Rw$ is a prefix of $Rv_1$.

- **C_1:** $q$ is a prefix of every word in $\text{NFA}(q)$
  - $q_1 = RXRX\ \underline{RXRX}(RX)^* = \text{NFA}(q_1)$

- **FO-rewritable**
  - $q_1 = RXRX\ \underline{RXRX}(RX)^* = \text{NFA}(q_1)$
$C_1$, $C_{1.5}$ and $C_2$ are decidable

$C_1$ : $q$ is a prefix of every word in NFA($q$)  

$\iff$ Whenever $q = u \cdot \underline{Rv} \cdot Rw$, $q$ is a prefix of $u \cdot \underline{Rv} \cdot Rv \cdot Rw$.

$C_2$ : $q$ is a factor of every word in NFA($q$)  

$\iff$ Whenever $q = u \cdot \underline{Rv} \cdot Rw$, $q$ is a factor of $u \cdot \underline{Rv} \cdot Rv \cdot Rw$. 
Proposition

Let \( q \) be a path query satisfying \( C_2 \). The following statements are equivalent:

1. \( \text{db} \) is a “yes”-instance for \( \text{CERTAINTY}(q) \); and
2. \( \exists c \) such that in all repairs, there exists a path accepted by \( \text{NFA}(q) \) starting in \( c \).

\( C_2 : \quad q \) is a factor of every word in \( \text{NFA}(q) \)

When \( q \) satisfies \( C_1, C_{1.5}, \) and \( C_2 \), item 2 can be checked in \( \text{FO}, \text{NL}, \) and \( \text{P} \) respectively.
Proposition

Let $q$ be a path query satisfying $C_2$. The following statements are equivalent:

1. $db$ is a “yes”-instance for $\text{CERTAINTY}(q)$; and
2. $\exists c$ such that in all repairs, there exists a path accepted by $\text{NFA}(q)$ starting in $c$.

\[ C_2: \quad q \text{ is a factor of every word in } \text{NFA}(q) \]

When $q$ satisfies $C_1$, $C_{1.5}$, and $C_2$, item 2 can be checked in $\text{FO}$, $\text{NL}$, and $\text{P}$ respectively.
Proposition

Let $q$ be a path query satisfying $C_2$. The following statements are equivalent:

1. $db$ is a “yes”-instance for \textsc{Certainty}(q); and
2. $\exists c$ such that in all repairs, there exists a path accepted by NFA($q$) starting in $c$.

$C_2$: $q$ is a factor of every word in NFA($q$)

When $q$ satisfies $C_1$, $C_{1.5}$, and $C_2$, item 2 can be checked in $\text{FO}$, $\text{NL}$, and $P$ respectively.
Proposition

Let $q$ be a path query satisfying $C_2$. The following statements are equivalent:

1. $db$ is a “yes”-instance for $CERTAINTY(q)$; and
2. $\exists c$ such that in all repairs, there exists a path accepted by $NFA(q)$ starting in $c$.

$C_2$: $q$ is a factor of every word in $NFA(q)$

When $q$ satisfies $C_1$, $C_{1.5}$, and $C_2$, item 2 can be checked in $FO$, $NL$, and $P$ respectively.
Proposition

Let $q$ be a path query satisfying $C_2$. The following statements are equivalent:

1. $db$ is a “yes”-instance for $CERTAINTY(q)$; and
2. $\exists c$ such that in all repairs, there exists a path accepted by $NFA(q)$ starting in $c$.

$C_2$: $q$ is a factor of every word in $NFA(q)$

When $q$ satisfies $C_1$, $C_{1.5}$, and $C_2$, item 2 can be checked in $FO$, $NL$, and $P$ respectively.
Proposition

Let $q$ be a path query satisfying $C_2$. The following statements are equivalent:

1. $db$ is a “yes”-instance for $\text{CERTAINTY}(q)$; and
2. $\exists c$ such that in all repairs, there exists a path accepted by $\text{NFA}(q)$ starting in $c$.

$C_2$: $q$ is a factor of every word in $\text{NFA}(q)$

When $q$ satisfies $C_1$, $C_{1.5}$, and $C_2$, item 2 can be checked in $\text{FO}$, $\text{NL}$, and $\text{P}$ respectively.
Proposition

Let $q$ be a path query satisfying $C_2$. The following statements are equivalent:

1. $db$ is a “yes”-instance for CERTAINTY($q$); and
2. $\exists c$ such that in all repairs, there exists a path accepted by NFA($q$) starting in $c$.

$C_2$: $q$ is a factor of every word in NFA($q$)

When $q$ satisfies $C_1$, $C_{1.5}$, and $C_2$, item 2 can be checked in FO, NL, and P respectively.
Hardness

Lemma

For a path query $q$,

- if $q$ violates $C_1$, then $\text{CERTAINTY}(q)$ is $\text{NL}$-hard;
- if $q$ violates $C_{1.5}$, then $\text{CERTAINTY}(q)$ is $\text{P}$-hard;
- if $q$ violates $C_2$, then $\text{CERTAINTY}(q)$ is $\text{coNP}$-hard.

via

- Reachability
- Monotone Circuit Value
- Unsatisfiability
P-hardness

$q = RXRYRY$ violates $C_{1.5}$

The output of $C$ is 0 iff $db$ contains a falsifying repair
## Complexity classification for Path Queries

<table>
<thead>
<tr>
<th>Complexity</th>
<th>Expression</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>coNP-complete</td>
<td>$q_4 = RXRX \ RYRY$</td>
<td>$RXRXRYRXRYRY \in \text{NFA}(q_4)$</td>
</tr>
<tr>
<td>P-complete</td>
<td>$q_3 = RX \ RYRY$</td>
<td></td>
</tr>
<tr>
<td>C$_2$</td>
<td>$q$ is a factor of every word in NFA($q$)</td>
<td></td>
</tr>
<tr>
<td>NL-complete</td>
<td>$q_2 = RX \ RY$</td>
<td>$RXRXRY \in \text{NFA}(q_2)$</td>
</tr>
<tr>
<td>C$_1.5$</td>
<td>Whenever $q = uRvRw$, $q$ is a factor of $uRvRvRw$; and whenever $q = uRv_1Rv_2Rw$ for consecutive occurrences of $R$, $v_1 = v_2$ or $Rw$ is a prefix of $Rv_1$.</td>
<td></td>
</tr>
<tr>
<td>C$_1$</td>
<td>$q$ is a prefix of every word in NFA($q$)</td>
<td></td>
</tr>
<tr>
<td>FO-rewritable</td>
<td>$q_1 = RXRX$</td>
<td>$RXRX(RX)^* = \text{NFA}(q_1)$</td>
</tr>
</tbody>
</table>
SJF rooted trees (and beyond)

- FO, P \ FO, coNP-complete
  - [KOW, PODS'24]

SJF paths

- FO, NL-complete, P-complete, coNP-complete
  - [KOW, PODS'21]

SJF

- FO, L-complete, coNP-complete
  - [KW, ICDT'19]

SJF

- FO, P \ FO, coNP-complete
  - [KW, PODS'15]

SJF two tables

- P, coNP-complete
  - [KP, IPL'12]

SJF simple keys

- P, coNP-complete
  - [KS, ICDT'14]

C\textsubscript{forest}

- FO
  - [FM, ICDT'05]

\(\alpha\)-acyclic

- FO, non-FO
  - [Wijsen, PODS'10]

theory

CAvSAT

- \* via SAT
  - [DK, SAT'19, ICDE'21]

EQUIP

- \* via BIP
  - [KPT, VLDB'13]

LinCQA

- PPJT via FO in \(O(N)\)
  - [FKOW, SIGMOD'23]

Conquesto

- SJF via FO
  - [AJL, CIKM'20]

ConQuer

- C\textsubscript{forest} via FO
  - [FM, SIGMOD'05]
\[ q = \exists x, y, z, w : R(x, y) \land R(y, z) \land X(z, w) = RRX \]

\[ q :\neg R(x, y), R(y, z), X(z, w) \]

\[ \text{no idea yet...} \]
\[ q = \exists x, y, z, w : R(x, y) \land R(y, z) \land X(z, w) = RRX \]
\[ q :\text{-} R(x, y), R(y, z), X(z, w) \]

\[
\begin{array}{cccc}
R & R & R & X \\
x & y & z & w \\
\bot & \end{array}
\]

\[ x \xrightarrow{R} y \xrightarrow{R} z \xrightarrow{X} w \]

no idea yet...
\[ q = \exists x, y, z, w : R(x, y) \land R(y, z) \land X(z, w) = RRX \]

\[ q : - R(x, y), R(y, z), X(z, w) \]

\[ x \xrightarrow{R} y \xrightarrow{R} z \xrightarrow{X} w \]

no idea yet...
variable mapping

$q_1$: $C(x, y, z), R(y, u_1, v_1), A(u_1), B(v_1), R(z, u_2, v_2), B(u_2), A(v_2)$

$q_2$: $C(x, y, z), R(y, u_1, v_1), A(u_1), B(v_1), R(z, u_2, v_2), A(u_2), B(v_2)$
variable mapping
variable mapping

$q_1 \leftarrow C(x, y, z), R(y, u_1, v_1), A(u_1), B(v_1), R(z, u_2, v_2), B(u_2), A(v_2).$
variable mapping

$q_1 \leftarrow C(x, y, z), R(y, u_1, v_1), A(u_1), B(v_1), R(z, u_2, v_2), B(u_2), A(v_2).$

$q_2 \leftarrow C(x, y, z), R(y, u_1, v_1), A(u_1), B(v_1), R(z, u_2, v_2), A(u_2), B(v_2).$
What about rewinding?

\[ q = R \xrightarrow{RX} R \xrightarrow{RRX} \]

\[ q^{R:z \leftrightarrow y} \]

variable mapping
What about rewinding?

$q = R \xrightarrow{RX} R \xrightarrow{RRX}$

$q \xrightarrow{R:z \rightarrow y}$

Variable mapping.
What about rewinding?

\[ q = R \quad \overset{RX}{\Rightarrow} \quad R \quad \overset{RRX}{\Rightarrow} \]

variable mapping:

\[ q^{R: z \mapsto y} \]
What about rewinding?

\[ q = R \xrightarrow{RX} R \xrightarrow{RRX} \]

\[ q \xrightarrow{R: z \mapsto y} \]

Variable mapping:

\[ C \xrightarrow{RX} \xrightarrow{RRX} \]

A \hspace{1cm} B \hspace{1cm} A

u_1 \hspace{1cm} v_1 \hspace{1cm} u_2 \hspace{1cm} v_2

\begin{align*}
q &= R \\
\text{replace} &\quad \text{with a “previous” word}
\end{align*}
\[ q^R: x_2 \mapsto x_1 \]

\[ q^R: x_1 \mapsto x_2 \]

\[ q^R: x_3 \mapsto x_1 \]
Classification on rooted trees

$C_2^\clubsuit$: for every $R\langle x \rangle$ and $R\langle y \rangle$ in $q$, there is a homomorphism from $q$ to either $q^{R:x\rightarrow y}$ or $q^{R:y\rightarrow x}$.

$q_1$ satisfies $C_2^\clubsuit$
Classification on rooted trees

$C_2^\clubsuit$: for every $R\langle x \rangle$ and $R\langle y \rangle$ in $q$, there is a homomorphism from $q$ to either

$q^{R:x\leftrightarrow y}$ or $q^{R:y\leftrightarrow x}$

$q_1$ satisfies $C_2^\clubsuit$
Classification on rooted trees

$C_2^\bullet$: for every $R\langle x \rangle$ and $R\langle y \rangle$ in $q$, there is a homomorphism from $q$ to either

$q^{R:x \mapsto y}$ or $q^{R:y \mapsto x}$

$q_2$ violates $C_2^\bullet$
Classification on rooted trees

\( C_2 \): for every \( R\langle x \rangle \) and \( R\langle y \rangle \) in \( q \), there is a homomorphism from \( q \) to either

\[ q^{R:x \rightarrow y} \text{ or } q^{R:y \rightarrow x} \]

Theorem

If \( q \) satisfies \( C_2 \), then \( \text{CERTAINTY}(q) \) is in \( P \), or otherwise \( \text{coNP}-\text{complete} \).
Classification on rooted trees

$C_1^\bullet$: for every $R\langle x \rangle$ and $R\langle y \rangle$ in $q$, there is a root homomorphism from $q$ to either $q^{R: x \rightarrow y}$ or $q^{R: y \rightarrow x}$.

$q_1$ satisfies $C_1^\bullet$

$q_3$ satisfies $C_1^\bullet$
Classification on rooted trees

$C_1^\bullet$: for every $R\langle x \rangle$ and $R\langle y \rangle$ in $q$, there is a root homomorphism from $q$ to either

$q^{R:x\rightarrow y}$ or $q^{R:y\rightarrow x}$

$q_1$ satisfies $C_1^\bullet$

$q_4 : \neg C_1^\bullet$, $C_2^\bullet$

Theorem

If $q$ satisfies $C_1^\bullet$, then $\text{CERTAINTY}(q)$ is in $\text{FO}$, or otherwise $\text{NL}$-hard.
Rooted trees generalize paths

**FO-rewritable C₁**: for every $R\langle x \rangle$ and $R\langle y \rangle$ in $q$, there is a root homomorphism from $q$ to either $q^{R:x\rightarrow y}$ or $q^{R:y\rightarrow x}$.
Rooted trees generalize paths

**FO-rewritable**  
$C_1^\clubsuit$: for every $R\langle x \rangle$ and $R\langle y \rangle$ in $q$, there is a \textit{root homomorphism} from $q$ to either $q^{R:x\mapsto y}$ or $q^{R:y\mapsto x}$
Rooted trees generalize paths

\[ \text{C}_2 : \text{ for every } R\langle x \rangle \text{ and } R\langle y \rangle \text{ in } q, \text{ there is a homomorphism from } q \text{ to either } q^{R:x \rightarrow y} \text{ or } q^{R:y \rightarrow x} \]

\[ \text{P} \]

\[ \text{C}_1 : \text{ for every } R\langle x \rangle \text{ and } R\langle y \rangle \text{ in } q, \text{ there is a root homomorphism from } q \text{ to either } q^{R:x \rightarrow y} \text{ or } q^{R:y \rightarrow x} \]

\[ \text{NL-hard} \]

\[ \text{FO-rewritable} \]
CoNP-complete

$\mathbf{C}_2^\bullet$: for every $R\langle x \rangle$ and $R\langle y \rangle$ in $q$, there is a homomorphism from $q$ to either $q^{R:x\leftarrow y}$ or $q^{R:y\leftarrow x}$

NL-hard

FO-rewritable

$\mathbf{C}_1^\bullet$: for every $R\langle x \rangle$ and $R\langle y \rangle$ in $q$, there is a root homomorphism from $q$ to either $q^{R:x\leftarrow y}$ or $q^{R:y\leftarrow x}$
Rooted trees generalize paths

**coNP-complete**

- $C_2^\bullet$: for every $R\langle x \rangle$ and $R\langle y \rangle$ in $q$, there is a homomorphism from $q$ to either $q^{R:x \rightarrow y}$ or $q^{R:y \rightarrow x}$

- $C_2$: $q = u R v R w$ is a factor of $u R v R v R w$

**P**

- $C_2^\bullet$: for every $R\langle x \rangle$ and $R\langle y \rangle$ in $q$, there is a homomorphism from $q$ to either $q^{R:x \rightarrow y}$ or $q^{R:y \rightarrow x}$

- $C_2$: $q = u R v R w$ is a factor of $u R v R v R w$

**NL-hard**

**FO-rewritable**

- $C_1^\bullet$: for every $R\langle x \rangle$ and $R\langle y \rangle$ in $q$, there is a root homomorphism from $q$ to either $q^{R:x \rightarrow y}$ or $q^{R:y \rightarrow x}$

- $C_1$: $q = u R v R w$ is a prefix of $u R v R v R w$
Good rooted trees are just “paths”

\[ C_2^{\bullet} : \text{for every } R\langle x \rangle \text{ and } R\langle y \rangle \text{ in } q, \text{ there is a homomorphism from } q \text{ to either} \]
\[ q^{\overset{R}{\to} x} \text{ or } q^{\overset{R}{\to} y} \]

**Definition:** \( R\langle x \rangle \preceq_q R\langle y \rangle \) if
- \( R\langle x \rangle \) is an ancestor of \( R\langle y \rangle \) in \( q \); or
- there is a homomorphism from \( q \) to \( q^{\overset{R}{\to} y} \)

**Proposition:** If \( q \) satisfies \( C_2^{\bullet} \), for every predicate name \( R \), the relation \( \preceq_q \) is a total preorder on all \( R \)-atoms.
Good rooted trees are just “paths”

\[ \mathcal{C}_2^\bullet: \text{for every } R\langle x \rangle \text{ and } R\langle y \rangle \text{ in } q, \text{ there is a homomorphism from } q \text{ to either } q^{R: x \leftrightarrow y} \text{ or } q^{R: y \leftrightarrow x} \]

**Definition:** \( R\langle x \rangle \preceq_q R\langle y \rangle \) if
- \( R\langle x \rangle \) is an ancestor of \( R\langle y \rangle \) in \( q \); or
- there is a homomorphism from \( q \) to \( q^{R: y \leftrightarrow x} \)

**Proposition:** If \( q \) satisfies \( \mathcal{C}_2^\bullet \), for every predicate name \( R \), the relation \( \preceq_q \) is a total preorder on all \( R \)-atoms.

\[
R\langle y \rangle \\
R\langle x \rangle \preceq_q \cdots \preceq_q R\langle u \rangle \\
R\langle z \rangle
\]
For good trees, checking one repair is all you need

\( C_\diamondsuit: \) for every \( R(x) \) and \( R(y) \) in \( q \), there is a \textit{homomorphism} from \( q \) to either

\[ q^{R:x \rightarrow y} \text{ or } q^{R:y \rightarrow x} \]

Problem: \( \text{CERTAINTY}(q) \), for a rooted tree query \( q \)

Input: a database \( db \)

Question: does \( \text{rep} \models q \) hold for every \( \text{rep} \) of \( db \)?

\[
\begin{align*}
\text{rep}_1 & \models q? \\
\text{rep}_2 & \models q? \\
\text{rep}_3 & \models q? \\
\cdots \\
\text{rep}_{2^n} & \models q?
\end{align*}
\]
For good trees, checking one repair is all you need

\[ C_2^\bullet : \text{for every } R\langle x \rangle \text{ and } R\langle y \rangle \text{ in } q, \text{ there is a homomorphism from } q \text{ to either} \]
\[ q^{R:x \mapsto y} \text{ or } q^{R:y \mapsto x} \]

Problem: \text{CERTAINTY}(q), \text{ for a rooted tree query } q

Input: a database \textbf{db}

Question: does \textbf{rep} \models q \text{ hold for every } \textbf{rep} \text{ of } \textbf{db} ?

\textbf{rep}_1 \models q? \quad \textbf{rep}_2 \models q? \quad \textbf{rep}_3 \models q? \quad \cdots \quad \textbf{rep}_{2^n} \models q?
For good trees, checking one repair is all you need

\( C_2^\clubsuit \): for every \( R\langle x \rangle \) and \( R\langle y \rangle \) in \( q \), there is a homomorphism from \( q \) to either

\[ q^{R:x \mapsto y} \text{ or } q^{R:y \mapsto x} \]

Problem: \( \text{CERTAINTY}(q) \), for a rooted tree query \( q \)

Input: a database \( db \)

Question: does \( rep \models q \) hold for every \( rep \) of \( db \)?

\( rep_1 \models q? \quad rep_2 \models q? \quad rep_3 \models q? \quad \cdots \quad rep^* \quad rep_{2^n} \models q? \)

Proposition: If \( q \) satisfies \( C_2^\clubsuit \), there exists some \( rep^* \) of \( db \) that depends on \( q \)
For good trees, checking one repair is all you need

\[ C_2^{\bullet}: \text{for every } R\langle x \rangle \text{ and } R\langle y \rangle \text{ in } q, \text{ there is a homomorphism from } q \text{ to either} \]
\[ q^{R: x \mapsto y} \text{ or } q^{R: y \mapsto x} \]

Problem: CERTAINTY(q), for a rooted tree query q

Input: a database db

Question: does rep \models q hold for every rep of db?

\[ \text{rep}_1 \models q? \quad \text{rep}_2 \models q? \quad \text{rep}_3 \models q? \quad \ldots \quad \text{rep}^* \quad \text{rep}_{2n} \models q? \]

Proposition: If q satisfies C_2^{\bullet}, there exists some rep* of db that depends on q such that

\[ \text{rep}^* \models q \]
For good trees, checking one repair is all you need

\[ C_2^\bullet: \text{for every } R\langle x \rangle \text{ and } R\langle y \rangle \text{ in } q, \text{ there is a homomorphism from } q \text{ to either} \]
\[ q^{R:x \mapsto y} \text{ or } q^{R:y \mapsto x} \]

Problem: \text{CERTAINTY}(q), \text{ for a rooted tree query } q

Input: a database \text{db}

Question: \text{does } rep \models q \text{ hold for every } \text{rep} \text{ of } \text{db} \text{?}

\[ \text{rep}_1 \models q? \quad \text{rep}_2 \models q? \quad \text{rep}_3 \models q? \quad \cdots \quad \text{rep}^* \quad \text{rep}_{2^n} \models q? \]

Proposition: If \( q \) satisfies \( C_2^\bullet \), there exists some \( \text{rep}^* \) of \( \text{db} \) that depends on \( q \) such that
\[ \text{rep}^* \models q \iff \text{rep} \models q \text{ for every } \text{rep} \text{ of } \text{db}. \]
For good trees, checking one repair is all you need

$C_2$: for every $R\langle x \rangle$ and $R\langle y \rangle$ in $q$, there is a homomorphism from $q$ to either

$q^{R: x \rightarrow y}$ or $q^{R: y \rightarrow x}$

Problem: CERTAINTY($q$), for a rooted tree query $q$
Input: a database $db$
Question: does $\text{rep} \models q$ hold for every $\text{rep}$ of $db$?

$\text{rep}_1 \models q? \quad \text{rep}_2 \models q? \quad \text{rep}_3 \models q? \quad \cdots \quad \text{rep}^* \quad \text{rep}_{2^n} \models q?$

Proposition: If $q$ satisfies $C_2$, there exists some $\text{rep}^*$ of $db$ that depends on $q$ such that

$\text{rep}^* \models q \iff \text{rep} \models q$ for every $\text{rep}$ of $db$.

Moreover, one such $\text{rep}^*$ can be found in $P$. 
Initialization Step: for every $c \in \text{adom}(db)$ and leaf variable or constant $u$ in $q$

$$\text{add } \langle c, u \rangle \text{ to } B \text{ if } \begin{cases} u = c \text{ is a constant,} \\ \text{or the label of variable } u \text{ in } q \text{ is either } \bot, \\ \text{or } L \text{ with } L(c) \in db. \end{cases}$$

Iterative Rule: for every $c \in \text{adom}(db)$ and atom $R(y, y_1, y_2, \ldots, y_n)$ in $q$

$$\text{add } \langle c, y \rangle \text{ to } B \text{ if the following formula holds:}$$

$$\exists \vec{d}: R(c, \vec{d}) \in db \land \forall \vec{d}: \left( R(c, \vec{d}) \in db \rightarrow \text{fact}(R(c, \vec{d}), y) \right),$$

where

$$\text{fact}(R(c, \vec{d}), y) = \left( \bigwedge_{1 \leq i \leq n} \langle d_i, y_i \rangle \in B \right) \lor \left( \bigvee_{R[x]<qR[y]} \text{fact}(R(c, \vec{d}), x) \right)$$

forward production

backward production

and $\vec{d} = \langle d_1, d_2, \ldots, d_n \rangle$. 

Xiating Ouyang
Consistent Query Answering
PhD Defense 65 / 71
Classification for rooted trees

<table>
<thead>
<tr>
<th>Classification</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>coNP-complete</strong></td>
<td><strong>C_2</strong>: for every ( R\langle x \rangle ) and ( R\langle y \rangle ) in ( q ), there is a homomorphism from ( q ) to either ( q^{R:x \mapsto y} ) or ( q^{R:y \mapsto x} )</td>
</tr>
<tr>
<td><strong>P</strong></td>
<td><strong>C_2</strong>: ( q = u \ R v \ R w ) is a factor of ( u \ R v \ R v \ R w )</td>
</tr>
<tr>
<td><strong>NL-hard</strong></td>
<td><strong>C_2</strong>: for every ( R\langle x \rangle ) and ( R\langle y \rangle ) in ( q ), there is a root homomorphism from ( q ) to either ( q^{R:x \mapsto y} ) or ( q^{R:y \mapsto x} )</td>
</tr>
<tr>
<td><strong>FO-rewritable</strong></td>
<td><strong>C_1</strong>: ( q = u \ R v \ R w ) is a prefix of ( u \ R v \ R v \ R w )</td>
</tr>
</tbody>
</table>

Can be extended to “(Berge-acyclic) Graph queries” ...
### Classification for rooted trees

<table>
<thead>
<tr>
<th>Complexity Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>coNP</strong>-complete</td>
<td>for every ( R\langle x \rangle ) and ( R\langle y \rangle ) in ( q ), there is a homomorphism from ( q ) to either ( q^{R:x \leftrightarrow y} ) or ( q^{R:y \leftrightarrow x} )</td>
</tr>
<tr>
<td><strong>P</strong></td>
<td>( C_2 ): ( q = u R v R w ) is a factor of ( u R v R v R w )</td>
</tr>
<tr>
<td><strong>NL</strong>-hard</td>
<td>( C_1 ): ( q = u R v R w ) is a prefix of ( u R v R v R w )</td>
</tr>
<tr>
<td><strong>FO-rewritable</strong></td>
<td>( C_1 ): for every ( R\langle x \rangle ) and ( R\langle y \rangle ) in ( q ), there is a root homomorphism from ( q ) to either ( q^{R:x \leftrightarrow y} ) or ( q^{R:y \leftrightarrow x} )</td>
</tr>
</tbody>
</table>

Can be extended to “(Berge-acyclic) Graph queries” . . .
Concluding remarks
Finding Consistent Answers from Inconsistent Data: Systems, Algorithms, and Complexity

- **kNN + missing values**
  - [Karlaš et al., VLDB’21]

- **kNN + FD**
  - P, coNP-complete
  - [FK, ICDT’22]

- **Bayes + missing**
  - (BOFK, submitted)

- **ML + dirty data**
  - ...

- **SJF path**
  - FO, P \ FO, coNP-complete
  - [KOW, PODS’24]

- **SJF rooted trees (and beyond)**
  - FO, NL-complete, P-complete, coNP-complete
  - [KOW, PODS’24]

- **kNN + FD**
  - P, coNP-complete
  - [FK, ICDT’22]

- **Bayes + missing**
  - (BOFK, submitted)

- **ML + dirty data**
  - ...

- **SJF + (intgrty. const.)**
  - ...

- **SJF + multiple keys**
  - FO, non-FO
  - [HW, PODS’22]

- **LinCQA+**
  - SJF acyclic FO in $O(N)$?
  - [KOW, PODS’21]

- **LinCQA**
  - PPJT via FO in $O(N)$
  - [FKOW, SIGMOD’23]

**Conjecture:** For every (union of) BCQ $q$, CERTAINTY$(q)$ is in P or coNP-complete.
## Finding Consistent Answers from Inconsistent Data: Systems, Algorithms, and Complexity

### kNN + missing values
- FO, P, coNP-complete
- [Karlaš et al., VLDB’21]

### FL path
- FO, NL-complete, P-complete, coNP-complete
- [KOW, PODS’24]

### SJF
- FO, L-complete, coNP-complete
- [KW, ICDT’19]

### kNN + FD
- P, coNP-complete
- [FK, ICDT’22]

### Bayes + missing
- (BOFK, submitted)

### ML + dirty data
- ...

<table>
<thead>
<tr>
<th>SJF + (intgrty. const.)</th>
<th>FO, non-FO?</th>
</tr>
</thead>
<tbody>
<tr>
<td>SJF + PK &amp; (unary)FK</td>
<td>FO, non-FO</td>
</tr>
<tr>
<td></td>
<td>[HW, PODS’22]</td>
</tr>
</tbody>
</table>

### SJF + multiple keys
- FO, non-FO
- [KW, PODS’20]

### SJF + ¬
- FO, non-FO
- [KW, PODS’18]

### LinCQA
- PPJT via FO in $O(N)$
- [FKOW, SIGMOD’23]

### Conjecture: For every (union of) BCQ $q$, CERTAINTY$(q)$ is in P or coNP-complete.
Finding Consistent Answers from Inconsistent Data: Systems, Algorithms, and Complexity

- **kNN + missing values**
  - FO, P \ FO, coNP-complete
  - [Karlaš et al., VLDB'21]

- **kNN + FD**
  - FO, coNP-complete
  - [FK, ICDT'22]

- **Bayes + missing**
  - ML + dirty data
  - (BOFK, submitted)

- **Graph query**
  - FO, P \ FO, coNP-complete?

- **SJF rooted trees (and beyond)**
  - FO, P \ FO, coNP-complete
  - [KOW, PODS'24]

- **SJF path**
  - FO, NL-complete, P-complete, coNP-complete
  - [KOW, PODS'21]

- **LinCQA**
  - PPJT via FO in $O(N)$
  - [KW, ICDT'19]

- **Conjecture:** For every (union of) BCQ $q$, CERTAINTY($q$) is in P or coNP-complete.
Finding Consistent Answers from Inconsistent Data: Systems, Algorithms, and Complexity

- **kNN + missing values**
  - FO, P \ FO, coNP-complete
  - [Karlaš et al., VLDB'21]

- **kNN + FD**
  - P, coNP-complete
  - [FK, ICDT'22]

- **Bayes + missing**
  - (BOFK, submitted)

- **ML + dirty data**
  - ...?

- **SJF**
  - SJF path
    - FO, NL-complete, P-complete, coNP-complete
    - [KOW, PODS'24]

- **SJF rooted trees (and beyond)**
  - FO, P \ FO, coNP-complete
  - [KOW, PODS'24]

- **graph query**
  - ... FO, P \ FO, coNP-complete?

- **kNN + FD**
  - FO, P \ FO, coNP-complete
  - [FK, ICDT'22]

- **Bayes + missing**
  - (BOFK, submitted)

- **ML + dirty data**
  - ...?

- **LinCQA**
  - PPJT via FO in $O(N)$
  - [FKOW, SIGMOD'23]

- **LinCQA+**
  - ... SJF acyclic FO in $O(N)$?

- **SJF + (intgry. const.)**
  - FO, non-FO?

- **SJF + PK & (unary)FK**
  - FO, non-FO
  - [HW, PODS'22]

- **SJF + multiple keys**
  - FO, non-FO
  - [KW, PODS'20]

Conjecture: For every (union of) BCQ $q$, CERTAINTY($q$) is in P or coNP-complete.
Finding Consistent Answers from Inconsistent Data: Systems, Algorithms, and Complexity

- **SJF** rooted trees (and beyond)
  - \( \text{FO, NL-complete, P-complete, coNP-complete} \)
  - [KOW, PODS'24]

- **SJF** path
  - \( \text{FO, L-complete, coNP-complete} \)
  - [KW, ICDT'19]

- **LinCQA**
  - PPJT via \( \text{FO in } O(N) \)
  - [FKOW, SIGMOD'23]

- **kNN** + missing values
  - \([\text{Karlaš et al., VLDB'21}]\)

- **Bayes** + missing
  - \([\text{BOFK, submitted}]\)

- **ML** + dirty data
  - \([\text{...?}]\)

- **LinCQA**+ \( \ldots \text{SJF acyclic FO in } O(N) \)
Finding Consistent Answers from Inconsistent Data: Systems, Algorithms, and Complexity

**SJF + missing values**
- FO, P \ FO, coNP-complete
  - [Karlaš et al., VLDB'21]

**kNN + FD**
- P, coNP-complete
  - [FK, ICDT'22]

**Bayes + missing**
- (BOFK, submitted)

**ML + dirty data**
- ...
  - ...

**SJF rooted trees (and beyond)**
- FO, P \ FO, coNP-complete
  - [KOW, PODS'24]

**graph query**
- ... FO, P \ FO, coNP-complete?

**SJF path**
- FO, NL-complete, P-complete, coNP-complete
  - [KOW, PODS'21]

**SJF**
- FO, L-complete, coNP-complete
  - [KW, ICDT’19]

**LinCQA**
- PPJT via FO in \(O(N)\)
  - [FKOW, SIGMOD'23]

**Conjecture:** For every (union of) BCQ \(q\), \(\text{CERTAINTY}(q)\) is in P or coNP-complete.

**SJF + (integrity. const.)**
- ...
  - ...

**SJF + PK & (unary)FK**
- FO, non-FO
  - [HW, PODS’22]

**SJF + multiple keys**
- FO, non-FO
  - [KW, PODS’20]

**SJF + ¬**
- FO, non-FO
  - [KW, PODS’18]

**LinCQA**
- ...
  - ...

**SJF + (integrity. const.)**
- ...
  - ...

**kNN + missing values**
- [Karlaš et al., VLDB’21]

**kNN + FD**
- P, coNP-complete
  - [FK, ICDT’22]

**Bayes + missing**
- (BOFK, submitted)

**ML + dirty data**
- ...
  - ...

**SJF rooted trees (and beyond)**
- FO, P \ FO, coNP-complete
  - [KOW, PODS'24]

**graph query**
- ... FO, P \ FO, coNP-complete?

**SJF path**
- FO, NL-complete, P-complete, coNP-complete
  - [KOW, PODS'21]

**SJF**
- FO, L-complete, coNP-complete
  - [KW, ICDT’19]

**LinCQA**
- PPJT via FO in \(O(N)\)
  - [FKOW, SIGMOD'23]
Finding Consistent Answers from Inconsistent Data: Systems, Algorithms, and Complexity

SJF

- **FO**, P- & coNP-complete
  - [KOW, PODS’24]

SJF rooted trees (and beyond)

- **FO**, P-complete, coNP-complete
  - [KOW, PODS’24]

SJF path

- **FO**, NL-complete, P-complete, coNP-complete
  - [KOW, PODS’24]

SJF

- **FO**, L-complete, coNP-complete
  - [KW, ICDT’19]

LinCQA

- PPJT via **FO** in \(O(N)\)
  - [FKOW, SIGMOD’23]

LinCQA+

- SJF acyclic **FO** in \(O(N)\)?

kNN + missing values

- [Karlaš et al., VLDB’21]

kNN + FD

- P, coNP-complete
  - [FK, ICDT’22]

Bayes + missing

- (BOFK, submitted)

ML + dirty data

- ...

Conjecture: For every (union of) BCQ \(q\), CERTAINTY(\(q\)) is in P or coNP-complete.

Xiating Ouyang

Consistent Query Answering

PhD Defense
Finding Consistent Answers from Inconsistent Data: Systems, Algorithms, and Complexity

\(k\text{NN} + \text{missing values}\)
- \([\text{Karlaš et al., VLDB'21}]\)

\(k\text{NN} + \text{FD}\)
- \(P, \text{coNP-complete}\)
- \([\text{FK, ICDT'22}]\)

\(\text{Bayes} + \text{missing}\)
- \((\text{BOFK, submitted})\)

\(\text{SJF path}\)
- \(\text{FO, NL-complete, P-complete, coNP-complete}\)
- \([\text{KOW, PODS'24}]\)

\(\text{SJF rooted trees (and beyond)}\)
- \(\text{FO, P \setminus FO, coNP-complete}\)
- \([\text{KOW, PODS'24}]\)

\(\text{SJF}\)
- \(\text{FO, L-complete, coNP-complete}\)
- \([\text{KW, ICDT'19}]\)

\(\text{LinCQA}\)
- \(\text{PPJT via FO in } O(N)\)
- \([\text{FKOW, SIGMOD'23}]\)

\(\text{LinCQA}\)
- \(\ldots \text{SJF acyclic FO in } O(N)\)
- \([\text{KOW, PODS'21}]\)

\(\text{SJF + PK & (unary)FK}\)
- \(\text{FO, non-FO}\)
- \([\text{HW, PODS'22}]\)

\(\text{SJF + multiple keys}\)
- \(\text{FO, non-FO}\)
- \([\text{KW, PODS'20}]\)

\(\text{SJF + (intgrty. const.)}\)
- \(\ldots \text{FO, non-FO?}\)

\(\text{ML + dirty data}\)
- \(\ldots ?\)

Conjecture: For every (union of) BCQ \(q\), \(\text{CERTAINTY}(q)\) is in \(P\) or \(\text{coNP-complete}\).
Finding Consistent Answers from Inconsistent Data: Systems, Algorithms, and Complexity

- **SJF** path
  - FO, NL-complete, P-complete, coNP-complete
  - [KOW, PODS’24]

- **LinCQA**
  - PPJT via FO in $O(N)$
  - [FKOW, SIGMOD’23]

- **Bayes + missing**
  - (BOFK, submitted)

- **SJF + (intgrty. const.)**
  - FO, non-FO?
  - [HW, PODS’22]

- **SJF + multiple keys**
  - FO, non-FO
  - [KW, PODS’20]

- **LinCQA**
  - ... SJF acyclic FO in $O(N)$?

- **SJF + PK & (unary)FK**
  - FO, non-FO
  - [HW, PODS’22]

- **kNN + missing values**
  - [Karlaš et al., VLDB’21]

- **kNN + FD**
  - P, coNP-complete
  - [FK, ICDT’22]

- **SJF rooted trees (and beyond)**
  - FO, P \ FO, coNP-complete
  - [KOW, PODS’24]

- **LinCQA+$^+$**
  - ... SJF acyclic FO in $O(N)$?

- **ML + dirty data**
  - ... ?

- **SJF + ¬FO**, non-FO
  - [KW, PODS’18]

- **SJF + multiple keys**
  - FO, non-FO
  - [KW, PODS’20]

Conjecture: For every (union of) BCQ $q$, CERTAINTY$(q)$ is in P or coNP-complete.

Xiating Ouyang
Consistent Query Answering
PhD Defense 68 / 71
Finding Consistent Answers from Inconsistent Data: Systems, Algorithms, and Complexity

<table>
<thead>
<tr>
<th>SJF rooted trees (and beyond)</th>
<th>$kNN + missing\ values$</th>
<th>$kNN + FD$</th>
<th>Bayes + missing</th>
</tr>
</thead>
<tbody>
<tr>
<td>FO, NL-complete, $P$, coNP-complete</td>
<td>[Karlaš et al., VLDB’21]</td>
<td>$P$, coNP-complete</td>
<td>(BOFK, submitted)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SJF path</th>
<th>graph query</th>
<th>$ML + dirty\ data$</th>
</tr>
</thead>
<tbody>
<tr>
<td>FO, NL-complete, $P$, coNP-complete</td>
<td>$\ldots$ FO, $P \setminus FO$, coNP-complete?</td>
<td>$\ldots$ ?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SJF</th>
<th>LinCQA$^+$</th>
<th>LinCQA</th>
</tr>
</thead>
<tbody>
<tr>
<td>$FO$, L-complete, coNP-complete</td>
<td>$\ldots$ SJF acyclic $FO$ in $O(N)$?</td>
<td>PPJT via $FO$ in $O(N)$</td>
</tr>
<tr>
<td>[KW, ICDT’19]</td>
<td></td>
<td>[FKOW, SIGMOD’23]</td>
</tr>
</tbody>
</table>

**Conjecture:** For every (union of) BCQ $q$, CERTAINTY($q$) is in $P$ or coNP-complete.
The Beauty of Bounded Gaps

A huge discovery about prime numbers—and what it means for the future of math.

BY JORDAN ELLEMBERG  MAY 22, 2013  •  4:44 PM
Thank YOU!

Uri Andrews, Jin-Yi Cai, Paris Koutris, Jignesh Patel, Jef Wijsen

Yixin Cao, Rocky K. C. Chang, AnHai Doan, Steve Foote, Wei-Chiao Hsu, Alekh Jindal, Phokion Kolaitis, Ren Mao, Jeff Naughton, Hung Ngo, Lowell Rausch, Abhishek Roy, Ning Tan, Angela Thorp, Kristen Tinetti, Bin Xu, Fan (Amy) Yang

Song Bian, Ting Cai, Bing An Chang, Xufeng Cai, Elvis Chang, Jiang Chang, Kaiyang Chen, Maggie Chen, Yiding Chen, Nick Corrado, Shaleen Deep, Austen Z. Fan, Yuhang Fan, Zhiwei Fan, Kevin Gaffney, Yue Gao, Evangelia Gergatsouli, Jinshan Gu, Xinyu Guan, Yang Guo, Ankur Goswami, Yilin He, Hengjing Huang, Shunyi Huang, Aarati Kakaraparthty, Yuping Ke, Fengan Li, Justin LiXie, Holdson Liang, Eric Lin, Derek Ma, Jeremy McMahan, Simiao Ren, Yue Shi, Kartik Sreenivasan, Xiaoxi Sun, Yuxin Sun, Remy Wang, Xiang Wang, Jingcheng Xu, Jie You, Peng Yu, Zhe Zeng, Jifan Zhang, Ling Zhang, Hangdong Zhao, Xingjian Zhen, Yi Zhou

Yufei Gao, Feng-Ying Ma, Hong Ouyang, Zhong-Zhan Ouyang, Yujia Peng, Weisheng Wang, Hao Wu, Bin Xia, Yifan Xia
Finding Consistent Answers from Inconsistent Data: Systems, Algorithms, and Complexity

- **$k$NN + missing values**
  - [Karlaš et al., VLDB'21]

- **$k$NN + FD**
  - \( P \), coNP-complete
  - [FK, ICDT'22]

- **Bayes + missing**
  - (BOFK, submitted)

- **ML + dirty data**
  - ...?

- **LinCQA**
  - PPJT via FO in \( O(N) \)
  - [FKOW, SIGMOD'23]

- **LinCQA +**
  - ...SJF acyclic FO in \( O(N) \)?

- **SJF path**
  - FO, NL-complete, P-complete, coNP-complete
  - [KOW, PODS'24]

- **SJF rooted trees (and beyond)**
  - FO, P \( \setminus \) FO, coNP-complete
  - [KOW, PODS'24]

- **SJF**
  - FO, L-complete, coNP-complete
  - [KW, ICDT’19]

- **graph query**
  - ... FO, P \( \setminus \) FO, coNP-complete?

- **SJF + (intgrty. const.)**
  - ... FO, non-FO?

- **SJF + PK & (unary)FK**
  - FO, non-FO
  - [HW, PODS'22]

- **SJF + multiple keys**
  - FO, non-FO
  - [KW, PODS'20]

- **SJF + ¬**
  - FO, non-FO
  - [KW, PODS'18]

- **Conjecture:** For every (union of) BCQ \( q \), CERTAINTY(\( q \)) is in \( P \) or coNP-complete.