A Unified Approach For Reverse Data Management

Neha Makhija
Northeastern University
(Joint work with Wolfgang Gatterbauer)

Data Management

But sometimes, results are
- unexpected
- undesirable
- not understandable
Reverse Data Management

But sometimes, results are
- unexpected
- undesirable
- not understandable

Then we need to reason about the input in terms of the output!
Reverse Data Management

“What change would it take to achieve output X?”

“The handling of transformations that perform actions on the input data, on behalf of desired outcomes in the output data”
Reverse Data Management: Example

“What change would it take to delete some values from the output?”

Deletion Propagation
- Minimize Source Side Effect
- Minimize View Side Effect

Dayal, Bernstein. On the correct translation of update operations on relational views, TODS 1982 [https://doi.org/10.1145/319732.319740]
Buneman, Khanna, Tan. On propagation of deletions and annotations through views, PODS 2002 [https://doi.org/10.1145/543613.543633]
Reverse Data Management: Example

“What minimum change would it take to ensure output is “fair”?”

Algorithmic Fairness
Reverse Data Management: Example

“What change would it take to minimize the generated provenance expression?”

Minimal Factorization
Reverse Data Management: Example

“What minimum change would it take to make tuple X counterfactual?”

Causal Responsibility
Our Goal

In other words, data complexity

Fixed Conjunctive Query (CQ) $Q$

+ Output (Set of Witnesses)

Arbitrary Database $D$

Solve Reverse Data Management Problem

If hard
- Solve exactly
- Approximate in PTIME

Find tractability border

Exploit structure in data
- Solve easy subclasses in PTIME

One unified algorithm
Simplest Reverse Data Management Problem?

Input \rightarrow \text{Conjunctive Query} \rightarrow \text{Output}
**Simplest Reverse Data Management Problem?**

```
Simplest Reverse Data Management Problem?
```

“What minimal change would it take to **delete the output**?”

- Diagnose Points of Failure
- Equivalent to Deletion Propagation with Source Side-Effects

• Reverse Data Management Problems
• **Our Focus: Resilience**
• Results
• Our Unified Approach
• Takeaways + Open Questions
Resilience: Example

\textbf{Sees}(person, movie) \textbf{Buys}(person, item) \textbf{Featured-In}(item, movie)
Resilience: Example

Query: What person *sees* a movie and *buys* an item *featured in* the movie?

Q(person, movie, item): \( \text{Sees}(\text{person}, \text{movie}), \text{B Buys}(\text{person}, \text{item}), \text{Featured-In}(\text{item, movie}) \)
Query:- What person sees a movie and buys an item featured in the movie?

Q(person, movie, item):- Sees(person, movie), Buys(person, item), Featured-In(item, movie)

Resilience: Example

<table>
<thead>
<tr>
<th>Sees</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Person</td>
<td>Movie</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Po</td>
<td>Moana</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pip</td>
<td>Moana</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pip</td>
<td>Mulan</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Buys</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Person</td>
<td>Item</td>
<td></td>
</tr>
<tr>
<td>Po</td>
<td>Iron</td>
<td></td>
</tr>
<tr>
<td>Pip</td>
<td>Iron</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Featured-In</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Movie</td>
<td>Item</td>
<td></td>
</tr>
<tr>
<td>Moana</td>
<td>Iron</td>
<td></td>
</tr>
<tr>
<td>Mulan</td>
<td>Iron</td>
<td></td>
</tr>
</tbody>
</table>
Query: What person sees a movie and buys an item featured in the movie?

Q(person, movie, item): Sees(person, movie), Buys(person, item), Featured-In(item, movie)

---

**Resilience: Example**

```
<table>
<thead>
<tr>
<th>person</th>
<th>movie</th>
<th>item</th>
</tr>
</thead>
<tbody>
<tr>
<td>Po</td>
<td>Moana</td>
<td>Iron</td>
</tr>
<tr>
<td>Pip</td>
<td>Moana</td>
<td>Iron</td>
</tr>
<tr>
<td>Pip</td>
<td>Mulan</td>
<td>Iron</td>
</tr>
</tbody>
</table>
```

---

**Output**

```
<table>
<thead>
<tr>
<th>person</th>
<th>movie</th>
<th>item</th>
</tr>
</thead>
<tbody>
<tr>
<td>Po</td>
<td>Moana</td>
<td>Iron</td>
</tr>
<tr>
<td>Pip</td>
<td>Moana</td>
<td>Iron</td>
</tr>
<tr>
<td>Pip</td>
<td>Mulan</td>
<td>Iron</td>
</tr>
</tbody>
</table>
```

---

**Domain values**

- S1: Moana
- S2: Moana
- S3: Mulan

**Tuples**

- f1: (Moana, Iron)
- f2: (Mulan, Iron)

---

**Q(Δ)**

```
<table>
<thead>
<tr>
<th>person</th>
<th>movie</th>
<th>item</th>
</tr>
</thead>
<tbody>
<tr>
<td>Po</td>
<td>Moana</td>
<td>Iron</td>
</tr>
<tr>
<td>Pip</td>
<td>Moana</td>
<td>Iron</td>
</tr>
<tr>
<td>Pip</td>
<td>Mulan</td>
<td>Iron</td>
</tr>
</tbody>
</table>
```
Resilience

Query: What person sees a movie and buys an item featured in the movie?

Q(person, movie, item): Sees(person, movie), Buys(person, item), Featured-In(item, movie)

Output

<table>
<thead>
<tr>
<th>Person</th>
<th>Movie</th>
<th>Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>Po</td>
<td>Moana</td>
<td>Iron</td>
</tr>
<tr>
<td>Pip</td>
<td>Moana</td>
<td>Iron</td>
</tr>
<tr>
<td>Pip</td>
<td>Mulan</td>
<td>Iron</td>
</tr>
</tbody>
</table>

s₁, s₂, s₃:

Parents:

- Sees: Po: s₁, Pip: s₂, Pip: s₃
- Buys: Po: s₁, Pip: s₂
- Featured-In: s₁, s₂, s₃
Resilience

Query: What person sees a movie and buys an item featured in the movie?

\[ Q(\text{person, movie, item}) : \text{Sees}(\text{person, movie}), \text{Buys}(\text{person, item}), \text{Featured-In}(\text{item, movie}) \]

<table>
<thead>
<tr>
<th>Sees</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Person</td>
<td>Movie</td>
</tr>
<tr>
<td>Po</td>
<td>Moana</td>
</tr>
<tr>
<td>Pip</td>
<td>Moana</td>
</tr>
<tr>
<td>Pip</td>
<td>Mulan</td>
</tr>
</tbody>
</table>

\[ Q_\Delta \]

<table>
<thead>
<tr>
<th>Buys</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Person</td>
<td>Item</td>
</tr>
<tr>
<td>Po</td>
<td>Iron</td>
</tr>
<tr>
<td>Pip</td>
<td>Iron</td>
</tr>
</tbody>
</table>

\[ s_1 b_1 f_1 s_2 b_2 f_2 s_3 \]

Output

<table>
<thead>
<tr>
<th>Person</th>
<th>Movie</th>
<th>Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>Po</td>
<td>Moana</td>
<td>Iron</td>
</tr>
<tr>
<td>Pip</td>
<td>Moana</td>
<td>Iron</td>
</tr>
<tr>
<td>Pip</td>
<td>Mulan</td>
<td>Iron</td>
</tr>
</tbody>
</table>

witness
Resilience

Query:- What person *sees* a movie and *buys* an item *featured in* the movie?

Q(person, movie, item):- *Sees*(person, movie), *Buys*(person, item), *Featured-In*(item, movie)

<table>
<thead>
<tr>
<th>Sees</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Person</td>
<td>Movie</td>
</tr>
<tr>
<td>Po</td>
<td>Moana</td>
</tr>
<tr>
<td>Pip</td>
<td>Moana</td>
</tr>
<tr>
<td>Pip</td>
<td>Mulan</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Buys</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Person</td>
<td>Item</td>
</tr>
<tr>
<td>Po</td>
<td>Iron</td>
</tr>
<tr>
<td>Pip</td>
<td>Iron</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Featured-In</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Movie</td>
<td>Item</td>
</tr>
<tr>
<td>Moana</td>
<td>Iron</td>
</tr>
<tr>
<td>Mulan</td>
<td>Iron</td>
</tr>
</tbody>
</table>

Output

<table>
<thead>
<tr>
<th>Person</th>
<th>Movie</th>
<th>Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>Po</td>
<td>Moana</td>
<td>Iron</td>
</tr>
<tr>
<td>Pip</td>
<td>Moana</td>
<td>Iron</td>
</tr>
<tr>
<td>Pip</td>
<td>Mulan</td>
<td>Iron</td>
</tr>
</tbody>
</table>

witness
Resilience

Query: What person sees a movie and buys an item featured in the movie?

\[ Q(\text{person}, \text{movie}, \text{item}) : \text{Sees}(\text{person}, \text{movie}), \text{Buys}(\text{person}, \text{item}), \text{Featured-In}(\text{item}, \text{movie}) \]

\[
Q_\Delta
\]

<table>
<thead>
<tr>
<th>Sees</th>
<th>Movies</th>
<th>Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person</td>
<td>Movie</td>
<td></td>
</tr>
<tr>
<td>Po</td>
<td>Moana</td>
<td></td>
</tr>
<tr>
<td>Pip</td>
<td>Moana</td>
<td></td>
</tr>
<tr>
<td>Pip</td>
<td>Mulan</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Buys</th>
<th>Person</th>
<th>Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>Po</td>
<td>Iron</td>
<td></td>
</tr>
<tr>
<td>Pip</td>
<td>Iron</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Featured-In</th>
<th>Movies</th>
<th>Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>Movie</td>
<td>Item</td>
<td></td>
</tr>
<tr>
<td>Moana</td>
<td>Iron</td>
<td></td>
</tr>
<tr>
<td>Mulan</td>
<td>Iron</td>
<td></td>
</tr>
</tbody>
</table>

Output:

<table>
<thead>
<tr>
<th>Person</th>
<th>Movie</th>
<th>Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>Po</td>
<td>Moana</td>
<td>Iron</td>
</tr>
<tr>
<td>Pip</td>
<td>Moana</td>
<td>Iron</td>
</tr>
<tr>
<td>Pip</td>
<td>Mulan</td>
<td>Iron</td>
</tr>
</tbody>
</table>
Resilience

Query: What person sees a movie and buys an item featured in the movie?

Q(person, movie, item): Sees(person, movie), Buys(person, item), Featured-In(item, movie)

Recall: Resilience = What minimal change would it take to delete the output?"
Resilience

Query: What person sees a movie and buys an item featured in the movie?
Q(person, movie, item): Sees(person, movie), Buys(person, item), Featured-In(item, movie)

Recall: Resilience = What minimal change would it take to delete the output?”
Resilience Complexity

Query: What person sees a movie and buys an item featured in the movie?
Q(person, movie, item): Sees(person, movie), Buys(person, item), Featured-In(item, movie)

Query: What person sees a movie and buys an item featured in the movie that is rare?
Q(person, movie, item): Sees(person, movie), Buys(person, item), Featured-In(item, movie), Rare(Item)

• Reverse Data Management Problems
• Our Focus: Resilience
• Results
• Our Unified Approach
• Takeaways + Open Questions
Resilience Complexity: Self-Join Free Queries

Resilience Set Semantics:

\[
\begin{align*}
\text{PTIME} & & \text{PTIME} & & \text{NPC} \\
\text{NP} & & \text{NPC} & & \text{NPC}
\end{align*}
\]

Resilience Bag Semantics:

\[
\begin{align*}
\text{PTIME} & & \text{NPC} & & \text{NPC}
\end{align*}
\]

Complexity Results for Self-Join Free Queries

Buneman, Khanna, Tan. On propagation of deletions and annotations through views, PODS 2002 [https://doi.org/10.1145/543613.543633](https://doi.org/10.1145/543613.543633)


<table>
<thead>
<tr>
<th>Unified Algorithm (Easy or Hard)</th>
<th>Prior Work (RES+RESP)</th>
<th>Our Approach (RES+RESP)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Self-Join Free PTIME Cases</strong></td>
<td>Flow based encodings specific to query [Freire+15]</td>
<td><strong>One Unified Algorithm</strong></td>
</tr>
<tr>
<td><strong>Self-Join Free NPC Cases:</strong> Exact Evaluation</td>
<td>No prior algorithm: brute force trivially</td>
<td>- Solves all cases</td>
</tr>
<tr>
<td><strong>Self-Join Free NPC Cases:</strong> Approximations</td>
<td>No prior algorithm</td>
<td>- All known PTIME cases terminate in PTIME</td>
</tr>
<tr>
<td>Queries with Self-Joins</td>
<td>Flow algorithms for some known PTIME queries [Freire+20]</td>
<td></td>
</tr>
</tbody>
</table>

Freire, Gatterbauer, Immerman, Meliou. New results for the complexity of resilience for binary conjunctive queries with self-joins, PODS 2020. [https://doi.org/10.1145/3375395.3387647](https://doi.org/10.1145/3375395.3387647)
## Unified Algorithm (Across Different Settings)

<table>
<thead>
<tr>
<th>Semantics</th>
<th>Prior Work</th>
<th>Our Work</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set Semantics</td>
<td>Self-Join Free Queries + Some Restricted SJ Queries</td>
<td>Set + Bag Semantics</td>
</tr>
</tbody>
</table>

### Queries

- Can leverage known FDs

### Functional Dependencies

- Take advantage of unspecified FDs in data

### Prior Work

- All UCQs

### Our Work
• Reverse Data Management Problems
• Our Focus: Resilience
• Results
• Our Unified Approach
  – Unified Algorithm
  – Unified Hardness Criterion
• Takeaways + Open Questions
Resilience as an Integer Linear Program (ILP)

\[ \text{min } c^T x \]

\[ Ax \geq b \]

\[ x \in \{0,1\}^n \]

**Objective vector**

**Constraint vector**

**Constraint matrix**

**Example:**

Find a minimizing solution that satisfies the following constraints.

\[ \text{min} \left[ x[s_1] + x[b_1] + x[f_1] + x[s_2] + x[b_2] + x[f_2] + x[s_3] \right] \]

\[ x[s_1] + x[b_1] + x[f_1] \geq 1 \]

\[ x[f_1] + x[s_2] + x[b_2] \geq 1 \]

\[ x[b_2] + x[f_2] + x[s_3] \geq 1 \]

\[ \forall \text{ tuples } t: \ x[t] \in \{0,1\} \]

**Output**

<table>
<thead>
<tr>
<th>Person</th>
<th>Movie</th>
<th>Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>Po</td>
<td>Moana</td>
<td>Iron</td>
</tr>
<tr>
<td>Pip</td>
<td>Moana</td>
<td>Iron</td>
</tr>
<tr>
<td>Pip</td>
<td>Mulan</td>
<td>Iron</td>
</tr>
</tbody>
</table>

Weights if applied can go in the objective.

Queries with self-joins are modelled the same way. Not all constraints must have the same arity.

Setting \( X[f_1] = X[b_2] = 1 \), and all other variables to 0 satisfies the ILP.

Optimal value = 2
Resilience as a Linear Program (LP)

\[
\begin{align*}
\text{min } & \ c^T x \\
\text{subject to } & \ Ax \geq b \\
& \ x \in \{0,1\}^n \\
& \ x = [0,1]^n
\end{align*}
\]

Example: \( Q_\Delta \)

\[
\begin{array}{ccc}
s_1 & b_1 & f_1 \\
s_2 & b_2 & f_2 \\
s_3 & & \\
\end{array}
\]

Output

<table>
<thead>
<tr>
<th></th>
<th>Po</th>
<th>Pip</th>
<th>Pip</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>Moana</td>
<td>Mulan</td>
<td>Moana</td>
</tr>
<tr>
<td>b1</td>
<td>Iron</td>
<td>Iron</td>
<td>Iron</td>
</tr>
<tr>
<td>f1</td>
<td>s2b2f2</td>
<td>s3b2f2</td>
<td>s1b1f1</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
\min [ x[s_1] + x[b_1] + x[f_1] + x[s_2] + x[b_2] + x[f_2] + x[s_3] ] \\
x[s_1] + x[b_1] + x[f_1] \geq 1 \\
x[f_1] + x[s_2] + x[b_2] \geq 1 \\
x[b_2] + x[f_2] + x[s_3] \geq 1 \\
\forall \text{ tuples } t: \ x[t] \in \{0,1\} \\
\forall \text{ tuples } t: 0 \leq x[t] \leq 1
\end{align*}
\]

Linear programs allow fractional solutions
We can get a lower bound in PTIME.
Unified Algorithm for Resilience

Theorem.
For all known PTIME cases of resilience, the **objective** of the LP relaxation is identical to the ILP.

But ILPs are NP-Hard!
Unified Algorithm for Resilience

For all known PTIME cases of Resilience, $\text{LP}=\text{ILP}$

But ILPs are NP-Hard!
Unified Algorithm for Resilience

Theorem. For all known PTIME cases of Resilience, LP=ILP

If LP=ILP, solvers can recover a solution efficiently!
When does LP = ILP?

Our PTIME constraint matrixes need not be balanced or Totally Unimodular

The PTIME cases go beyond these known criterion!

For all known PTIME cases of Resilience, LP=ILP

Alexander Schrijver

THEORY OF LINEAR AND INTEGER PROGRAMMING

Combinatorial Optimization

Polyhedra and Efficiency

Volume A-C
When does LP = ILP?

Theorem.
For all known PTIME cases of Resilience, LP=ILP

Query → Build Resilience ILP → Database

Flow graph
Show correspondence to a flow graph

Use classical Max-Flow Min-Cut Theorem to show LP = ILP
• Reverse Data Management Problems
• Our Focus: Resilience
• Results
• Our Unified Approach
  – Unified Algorithm
  – Unified Hardness Criterion
• Takeaways + Open Questions
Unified Approach to prove Hardness

Vertex Cover → Resilience(Q)

Reduction Gadget ?

Arbitrary graph $G$

$VC(G) = k$

Corresponding database $D$

Resilience(Q, D) = f(k)
Unified Approach to prove Hardness

Vertex Cover \rightarrow \text{Resilience}(Q)

\text{Reduction Gadget ?}

Single Edge

\begin{align*}
\text{Edge Gadget} \quad & \quad \text{Corresponding database } D \\
\end{align*}

\begin{align*}
\text{Sees} & \\
\begin{array}{c}
s_1 \\
s_2 \\
s_3 \\
\end{array} & \quad \begin{array}{c}
\ldots \\
\ldots \\
\ldots \\
\end{array} & \quad \begin{array}{c}
\ldots \\
\ldots \\
\ldots \\
\end{array} \\

\text{Featured-In} & \\
\begin{array}{c}
f_1 \\
f_2 \\
f_3 \\
\end{array} & \quad \begin{array}{c}
\ldots \\
\ldots \\
\ldots \\
\end{array} & \quad \begin{array}{c}
\ldots \\
\ldots \\
\ldots \\
\end{array} \\

\text{Buys} & \\
\begin{array}{c}
b_1 \\
b_2 \\
b_3 \\
\end{array} & \quad \begin{array}{c}
\ldots \\
\ldots \\
\ldots \\
\end{array} & \quad \begin{array}{c}
\ldots \\
\ldots \\
\ldots \\
\end{array}
\end{align*}

\begin{align*}
\text{VC}(G) = k \\
\text{Resilience}(Q, D) = f(k)
\end{align*}
Unified Approach to prove Hardness

Vertex Cover → Resilience(Q)

Reduction Gadget ?

Single Edge

Edge Gadget

Independent Join Path

We prove: If a query forms an “Independent Join Path” → it is NPC (conjecture Freire+20)

$VC(G) = k$

Corresponding database $D$

Data Hypergraph

Resilience(Q, D) = f(k)
Unified Approach to prove Hardness

What is an Independent Join Path?

Database under query $Q$, with endpoints, with 5 *testable* properties:

1. Data hypergraph is connected

![Data Hypergraph](image-url)
Unified Approach to prove Hardness

What is an Independent Join Path?

Database under query $Q$, with endpoints, with 5 testable properties:

1. Data hypergraph is connected
2. Database is reduced

Data Hypergraph
Unified Approach to prove Hardness

What is an Independent Join Path?

Database under query $Q$, with endpoints, with 5 *testable* properties:

1. Data hypergraph is connected
2. Database is reduced
3. Endpoints are “valid”
Unified Approach to prove Hardness

What is an Independent Join Path?

Database under query $Q$, with endpoints, with 5 testable properties:

1. Data hypergraph is connected
2. Database is reduced
3. Endpoints are “valid”
4. OR – property
5. Composability

“Key” properties:
- Semantically defined
- I will just show intuition
Unified Approach to prove Hardness

Key Property #1: OR property

Pick at least a or b

Do not delete at least S1 or S3
Unified Approach to prove Hardness

**Key Property #1: OR property**

- Pick at least a or b
- VC = ??
  - Violates constraints

- Edge $\rightarrow$ IJP

- Do not delete at least S1 or S3
- RES = 3
  - Violates minimality

- Edge $\rightarrow$ IJP
Unified Approach to prove Hardness

Key Property #1: OR property

- Pick at least a or b
- VC = 1

Edge → IJP
- Do not delete at least S1 or S3

RES = 2
Unified Approach to prove Hardness

Key Property #1: OR property

Pick at least $a$ or $b$

VC = 1

Do not delete at least $S_1$ or $S_3$

RES = 2
Unified Approach to prove Hardness

**Key Property #1: OR property**

Pick at least a or b

VC = 2

Do not delete at least S1 or S3

RES = 2
Unified Approach to prove Hardness

**Key Property #2: Composability of IJPs**

Composing two IJPs should not lead to additional witnesses
Our Goal

Using the complete criterion for IJP, can we build a principled way to find IJPs?

Query

Automatic Gadget Finder

IJP (if exists)
Unified Approach to prove Hardness

Using the complete criterion for IJP, we can build a principled way to find IJPs

**Query, Domain Size of IJP**

**Automatic Gadget Finder**
Search through an exponential space – “guess” an IJP

**Verify IJP:**
- Check IJP has OR property
- Check IJP composes

**Certificate of Hardness**
Unified Approach to prove Hardness

Using the complete criterion for IJP, we can build a principled way to find IJPs

**Query, Domain Size of IJP**

---

**Automatic Gadget Finder**

Search through an exponential space – “guess” an IJP

**Verify IJP:**

- Check IJP has OR property
- Find size \( k \) resilience set (\( NP \))
- Find no size \( k-1 \) resilience set (\( Co-NP \))

---

**Certificate of Hardness**

More on size in just a bit
Using the complete criterion for IJP, we can build a principled way to find IJPs.

**Unified Approach to prove Hardness**

**Query, Domain Size of IJP**

**Automatic Gadget Finder**

Search through an exponential space – “guess” an IJP

Verify IJP:

- Check IJP has OR property
  - Find size $k$ resilience set ($\text{NP}$)
  - Find no size $k$-1 resilience set ($\text{Co-NP}$)

- Check IJP composes
  - It suffices to check just 3 join paths compose

**Certificate of Hardness**

More on size in just a bit
Unified Approach to prove Hardness

Using the complete criterion for IJP, we can build a principled way to find IJPs

Certificate of Hardness

Query, Domain Size of IJP

More on size in just a bit

Automatic Hardness Finder

Disjunctive Logic Program

\( \Sigma^2_p \)

Declarative

\( \chi \lor y :\neg z \)

Certificate of Hardness
Finding IJPs with DLP: 5 New Hardness Gadgets

- Using the **Automatic IJP Generator** we proved 5 queries hard (out of 7 previously **open** from Freire+20)
  
  \[ q^{S}_{3cc}: R(x,y), R(y,z), R(w,z), S(w,z) \]

  \[ q^{S_{xy}}_{3perm-R}: S(x,y), R(x,y), B(y), R(y,z), R(z,y) \]

  \[ q^{AS_{xy}}_{3perm-R}: A(x), S(x,y), R(x,y), R(y,z), R(z,y) \]

  \[ q^{S_{xy}}_{3perm-R}: S(x,y), R(x,y), R(y,z), R(z,y), C(z) \]

- Can recover all previous hardness results + find new ones!
Dichotomy Conjectures for Resilience

**Theorem.**

\[ \text{IJP} \rightarrow \text{NPC} \]

**Theorem.**

\[ \text{IJP} \leftrightarrow \text{NPC} \text{ for SJ-Free Queries} \]

**Conjecture. [Hardness]**

\[ \text{IJP} \leftrightarrow \text{NPC} \]

**Conjecture. [Hardness] Corollary**

\[ \text{NPC} \rightarrow \text{DLP finds a hardness proof} \]

**Conjecture. [Hardness]**

\[ \text{IJP} \rightarrow \text{IJP of domain size } \leq 7 \times \text{var}(Q) \]

**Conjecture. [PTIME]**

\[ \not \exists \text{IJP} \rightarrow \text{LP} = \text{ILP} \]

**Conjecture. [PTIME]**

\[ \not \exists \text{IJP} \rightarrow \text{There is a flow graph that encodes resilience} \]
Takeaways

• One **unified** algorithm, only need to prove PTIME
• One **unified** hardness criterion
  – Automatic search

Open Problems

• Which RDM problems can we solve with this unified approach?
  – Resilience
  – Causal Responsibility
  – Minimal Factorization of Provenance of CQs
  – ...... **Claim: many more**

Many more details, proofs, experiments, approximations: