Open-World Finite Query Answering with Number Restrictions

Antoine Amarilli\textsuperscript{1,2}, Michael Benedikt\textsuperscript{2}

\textsuperscript{1}Télécom ParisTech, Paris, France
\textsuperscript{2}University of Oxford, Oxford, United Kingdom

July 7, 2015
Open-world query answering (QA)

- **Open-world** query answering:
  - Relational instance $I$ (ground facts), correct but incomplete
  - Boolean conjunctive query $q$
  - Consider all possible completions $J \supseteq I$
  - Is $q$ certain, i.e., holds on all completions? If yes, $I \models_{\text{unr}} q$
Constraints

- Impose constraints on the possible completions
  - Inclusion dependencies: some facts imply more facts:

\[
\forall x d \ Employee(x, d) \Rightarrow \exists y \ Advises(x, y, d)
\]

\[
Employee[1, 2] \subseteq Advises[1, 3]
\]
Impose constraints on the possible completions

- Inclusion dependencies: some facts imply more facts:
  \[ \forall x d \ Employee(x, d) \Rightarrow \exists y \ Advises(x, y, d) \]
  \[ Employee[1, 2] \subseteq Advises[1, 3] \]
  \[ \rightarrow \text{Unary inclusion dependencies (UIDs): one shared position} \]
Constraints

- Impose **constraints** on the possible completions
  - **Inclusion dependencies**: some facts imply more facts:
    \[ \forall x d \text{ Employee}(x, d) \Rightarrow \exists y \text{ Advises}(x, y, d) \]
    Employee[1, 2] \subseteq Advises[1, 3]
  - **Unary** inclusion dependencies (UIDs): one shared position
  - **Functional dependencies** (FDs): uniqueness constraints:
    \[ \forall xx' y d \text{ Advises}(x, y, d) \land \text{Advises}(x', y, d) \rightarrow x = x' \]
    Advises[2, 3] \rightarrow \text{Advises}[1]
Impose **constraints** on the possible completions

- **Inclusion dependencies:** some facts imply more facts:
  \( \forall x d \ \text{Employee}(x, d) \Rightarrow \exists y \ \text{Advises}(x, y, d) \)
  \( \text{Employee}[1, 2] \subseteq \text{Advises}[1, 3] \)
  \( \rightarrow \) **Unary** inclusion dependencies (UIDs): one shared position

- **Functional dependencies (FDs):** uniqueness constraints:
  \( \forall xx'yd \ \text{Advises}(x, y, d) \land \text{Advises}(x', y, d) \rightarrow x = x' \)
  \( \text{Advises}[2, 3] \rightarrow \text{Advises}[1] \)

  \( \rightarrow \) Consider all \( J \supseteq I \) that **satisfy the constraints**

  \( \rightarrow \) Is \( q \) **certain** on all such completions? If yes, \( I, \Sigma \models_{\text{unr}} q \)
Finite vs infinite

Instance: List of adviser–advisee pairs: Advises\((a, b)\), \ldots

UID: Each advisee advises someone
Advises[2] \subseteq Advises[1]

FD: Each advisee has only one adviser

Query: Are all advisers themselves advised by someone?
Finite vs infinite

**Instance:** List of adviser–advisee pairs: \( \text{Advises}(a, b), \ldots \)

**UID:** Each advisee advises someone
\[
\text{Advises}[2] \subseteq \text{Advises}[1]
\]

**FD:** Each advisee has only one adviser
\[
\text{Advises}[2] \rightarrow \text{Advises}[1]
\]

**Query:** Are all advisers themselves advised by someone?

- **\( q \) is not certain:**
  \[
  \text{Advises}(a, b), \text{Advises}(b, b_2), \text{Advises}(b_2, b_3), \ldots
  \]

- **\( q \) is certain on all finite completions**
Finite vs infinite

Instance: List of adviser–advisee pairs: Advises\((a, b)\), \ldots

UID: Each advisee advises someone
Advises\([2]\) \subseteq Advises\([1]\)

FD: Each advisee has only one adviser
Advises\([2]\) \rightarrow Advises\([1]\)

Query: Are all advisers themselves advised by someone?

\- \textit{q is not certain:}
\begin{align*}
\text{Advises}(a, b), \text{Advises}(b, b_2), \text{Advises}(b_2, b_3), \ldots
\end{align*}

\- \textit{q is certain on all finite completions}

\rightarrow \text{Finite QA: only finite completions. If yes, } I, \Sigma \models_{\text{fin}} q
Finite vs infinite

Instance: List of adviser–advisee pairs: Advises\((a, b)\), \ldots

UID: Each advisee advises someone

FD: Each advisee has only one adviser

Query: Are all advisers themselves advised by someone?

- \(q\) is not certain:
  Advises\((a, b)\), Advises\((b, b_2)\), Advises\((b_2, b_3)\), \ldots

- \(q\) is certain on all finite completions

→ Finite QA: only finite completions. If yes, \(I, \Sigma \models_{\text{fin}} q\)

→ \(\Sigma\) finitely controllable: \(\forall I \forall q, I, \Sigma \models_{\text{fin}} q\) iff \(I, \Sigma \models_{\text{unr}} q\)
Table of contents

1. Introduction
2. Context and result
3. Proof ideas
4. Conclusion
(Finite) open-world query answering is **undecidable** for IDs and FDs [Calì et al., 2003]
Existing results

- (Finite) open-world query answering is **undecidable** for IDs and FDs [Calì et al., 2003]
- IDs alone are **finitely controllable** [Rosati, 2006]
Existing results

- (Finite) open-world query answering is undecidable for IDs and FDs [Calì et al., 2003]
- IDs alone are finitely controllable [Rosati, 2006]
- Infinite open-world query answering is decidable for UIDs and FDs [Calì et al., 2012] but not finitely controllable (see above; [Rosati, 2006])
(Finite) open-world query answering is undecidable for IDs and FDs [Calì et al., 2003]

IDs alone are finitely controllable [Rosati, 2006]

Infinite open-world query answering is decidable for UIDs and FDs [Calì et al., 2012] but not finitely controllable (see above; [Rosati, 2006])

(Finite) open-world query answering is decidable for IDs and FDs with relations of arity $\leq 2$ [Pratt-Hartmann, 2009, Ibáñez-García et al., 2014]
Problem and main result

- Study finite QA for UIDs and FDs on arbitrary signatures
- Is it decidable? What is the complexity?
Problem and main result

- Study **finite QA** for **UIDs and FDs** on arbitrary signatures
- Is it **decidable**? What is the **complexity**?

**Theorem**

*Finite open-world query answering for UIDs and FDs has \textit{PTIME} data complexity and \textit{NP-complete} combined complexity.*
Main idea

- Find which UIDs and FDs are implied over finite instances
  → This is PTIME [Cosmadakis et al., 1990]
  → It differs from unrestricted implication

- Finite closure: add all finitely implied dependencies
Main idea

- Find which UIDs and FDs are implied over finite instances
  - This is PTIME [Cosmadakis et al., 1990]
  - It differs from unrestricted implication

- Finite closure: add all finitely implied dependencies
  - Taking the finite closure ensure finite controllability?
Main idea

- Find which UIDs and FDs are implied over finite instances
  -> This is PTIME [Cosmadakis et al., 1990]
  -> It differs from unrestricted implication

- Finite closure: add all finitely implied dependencies
  -> Taking the finite closure ensure finite controllability?

**Theorem**

For any instance $I$, conjunctive query $q$, UIDs and FDs $\Sigma$, letting $\Sigma^*$ be the finite closure of $\Sigma$, we have $I, \Sigma \models_{\text{fin}} q$ iff $I, \Sigma^* \models_{\text{unr}} q$. 
# Table of contents

1. Introduction
2. Context and result
3. Proof ideas
4. Conclusion
Rephrasing the result

1. Assume that the UIDs and FDs $\Sigma$ are finitely closed
Rephrasing the result

- Assume that the UIDs and FDs $\Sigma$ are **finitely closed**
- Assume that an infinite completion $J$ satisfies $\Sigma$ but not $q$
Assume that the UIDs and FDs $\Sigma$ are **finitely closed**

Assume that an infinite completion $J$ satisfies $\Sigma$ but not $q$

\[\Rightarrow\] Construct a finite completion $J'$ satisfying $\Sigma$ but not $q$:

- Start with $I$
- Follow $J$ and add facts to satisfy the UIDs of $\Sigma$...
- ... using finitely many elements (reuse them)
- ... but respecting the FDs of $\Sigma$
- ... and without making $q$ inadvertently true
Constructing the finite completion

Make **assumptions**, we lift them from bottom to top:
Constructing the finite completion

Make assumptions, we lift them from bottom to top:

5. Assume that the query $q$ is **acyclic**
   e.g., $\exists xy \; R(x) S(x, y)$ but not $\exists xyz \; R(x, y) S(y, z) T(z, x)$
Constructing the finite completion

Make assumptions, we lift them from bottom to top:

5. Assume that the query $q$ is acyclic
   e.g., $\exists xy \ R(x) S(x, y)$ but not $\exists xyz \ R(x, y) S(y, z) T(z, x)$

4. Assume that there are only unary FDs
Constructing the finite completion

Make assumptions, we lift them from bottom to top:

5. Assume that the query \( q \) is acyclic
   e.g., \( \exists xy \ R(x) S(x, y) \) but not \( \exists xyz \ R(x, y) S(y, z) T(z, x) \)

4. Assume that there are only unary FDs
   e.g., \( R[1] \rightarrow R[2] \) but not \( R[1, 3] \rightarrow R[2] \)

3. Assume a certain reversibility property on the UIDs and FDs
Constructing the finite completion

Make **assumptions**, we lift them from bottom to top:

5. Assume that the query $q$ is **acyclic**
   e.g., $\exists xy \ R(x)\, S(x, y)$ but not $\exists xyz \ R(x, y)\, S(y, z)\, T(z, x)$

4. Assume that there are only **unary FDs**

3. Assume a certain **reversibility property** on the UIDs and FDs

2. Assume we have the **trivial query** $\bot$
Constructing the finite completion

Make assumptions, we lift them from bottom to top:

5. Assume that the query $q$ is acyclic
e.g., $\exists xy \ R(x) S(x, y)$ but not $\exists xyz \ R(x, y) S(y, z) T(z, x)$

4. Assume that there are only unary FDs

3. Assume a certain reversibility property on the UIDs and FDs

2. Assume we have the trivial query $\bot$

1. Assume that the relations all have arity two
Constructing the finite completion

Make assumptions, we lift them from bottom to top:

5. Assume that the query $q$ is acyclic
   e.g., $\exists xy R(x)S(x, y)$ but not $\exists xyz R(x, y)S(y, z)T(z, x)$

4. Assume that there are only unary FDs

3. Assume a certain reversibility property on the UIDs and FDs

2. Assume we have the trivial query $\bot$

1. Assume that the relations all have arity two

0. Solve the problem!
Constructing the finite completion

Make **assumptions**, we lift them from bottom to top:

5. Assume that the query $q$ is **acyclic**
   e.g., $\exists xy \ R(x)S(x, y)$ but not $\exists xyz \ R(x, y)S(y, z)T(z, x)$

4. Assume that there are only **unary FDs**
   e.g., $R[1] \to R[2]$ but not $R[1, 3] \to R[2]$

3. Assume a certain **reversibility property** on the UIDs and FDs

2. Assume we have the **trivial query** $\perp$

1. Assume that the relations all have **arity two**

0. Solve the problem!
   $\rightarrow$ Exemplified soon
Constructing the finite completion

Make **assumptions**, we lift them from bottom to top:

5. Assume that the query $q$ is acyclic
   e.g., $\exists xy R(x) S(x, y)$ but not $\exists xyz R(x, y) S(y, z) T(z, x)$

4. Assume that there are only unary FDs

3. Assume a certain **reversibility property** on the UIDs and FDs

2. Assume we have the **trivial query** $\bot$

1. Assume that the relations all have **arity two**
   $\rightarrow$ Reuse elements or extend previous case – skipped

0. Solve the problem!
   $\rightarrow$ Exemplified soon
Constructing the finite completion

Make assumptions, we lift them from bottom to top:

5. Assume that the query $q$ is acyclic
e.g., $\exists xy \ R(x) \ S(x, y)$ but not $\exists xyz \ R(x, y) \ S(y, z) \ T(z, x)$

4. Assume that there are only unary FDs

3. Assume a certain reversibility property on the UIDs and FDs

2. Assume we have the trivial query $\bot$
   → Exemplified soon

1. Assume that the relations all have arity two
   → Reuse elements or extend previous case – skipped

0. Solve the problem!
   → Exemplified soon
Constructing the finite completion

Make **assumptions**, we lift them from bottom to top:

5. Assume that the query $q$ is acyclic
   e.g., $\exists xy\ R(x)S(x, y)$ but not $\exists xyz\ R(x, y)S(y, z)T(z, x)$

4. Assume that there are only **unary FDs**

3. Assume a certain **reversibility property** on the UIDs and FDs
   $\rightarrow$ Decomposition of UIDs based on [Cosmadakis et al., 1990] – skipped

2. Assume we have the **trivial query** $\perp$
   $\rightarrow$ Exemplified soon

1. Assume that the relations all have **arity two**
   $\rightarrow$ Reuse elements or extend previous case – skipped

0. Solve the problem!
   $\rightarrow$ Exemplified soon
Constructing the finite completion

Make **assumptions**, we lift them from bottom to top:

5. Assume that the query $q$ is **acyclic**
   e.g., $\exists xy R(x)S(x, y)$ but not $\exists xyz R(x, y)S(y, z)T(z, x)$

4. Assume that there are only **unary FDs**
   \(\rightarrow\) **Recombination** of elements in patterns when reusing – skipped

3. Assume a certain **reversibility property** on the UIDs and FDs
   \(\rightarrow\) **Decomposition** of UIDs based on [Cosmadakis et al., 1990] – skipped

2. Assume we have the **trivial query** $\bot$
   \(\rightarrow\) **Exemplified soon**

1. Assume that the relations all have **arity two**
   \(\rightarrow\) **Reuse** elements or **extend** previous case – skipped

0. Solve the problem!
   \(\rightarrow\) **Exemplified soon**
Constructing the finite completion

Make assumptions, we lift them from bottom to top:

5. Assume that the query $q$ is acyclic
   e.g., $\exists xy \ R(x) S(x, y)$ but not $\exists xyz \ R(x, y) S(y, z) T(z, x)$
   $\rightarrow$ Product with group of high girth inspired by [Otto, 2002] – skipped

4. Assume that there are only unary FDs
   $\rightarrow$ Recombination of elements in patterns when reusing – skipped

3. Assume a certain reversibility property on the UIDs and FDs
   $\rightarrow$ Decomposition of UIDs based on [Cosmadakis et al., 1990] – skipped

2. Assume we have the trivial query $\perp$
   $\rightarrow$ Exemplified soon

1. Assume that the relations all have arity two
   $\rightarrow$ Reuse elements or extend previous case – skipped

0. Solve the problem!
   $\rightarrow$ Exemplified soon
0. Satisfying UIDs and UFDs in arity-two
0. Satisfying UIDs and UFDs in arity-two
0. Satisfying UIDs and UFDs in arity-two
0. Satisfying UIDs and UFDs in arity-two

**UIDs:**
- $\mathbb{R}[2] \subseteq \mathbb{S}[1]$
- $\mathbb{S}[2] \subseteq \mathbb{R}[1]$

**UFDs:**
- $\mathbb{R}[1] \leftrightarrow \mathbb{R}[2]$
- $\mathbb{S}[1] \leftrightarrow \mathbb{S}[2]$

Diagram:
- Two sets, $\mathbb{R}$ and $\mathbb{S}$, with arrows indicating the relations between them.
- Nodes labeled $a$, $b$, $c$, $d$, and $e$ with arrows connecting them in a specific pattern.
0. Satisfying UIDs and UFDs in arity-two

**UIDs:**
- \( S[2] \subseteq R[1] \)

**UFDs:**
- \( R[1] \leftrightarrow R[2] \)
- \( S[1] \leftrightarrow S[2] \)
0. Satisfying UIDs and UFDs in arity-two

**UIDs:**
- $R[2] \subseteq S[1]$
- $S[2] \subseteq R[1]$

**UFDs:**
- $R[1] \leftrightarrow R[2]$
- $S[1] \leftrightarrow S[2]$

![Diagram with nodes R, S, a, b, c, d, e and arrows indicating relationships between them]
0. Satisfying UIDs and UFDs in arity-two

**UIDs:**
- $R[2] \subseteq S[1]$
- $S[2] \subseteq R[1]$

**UFDs:**
- $R[1] \leftrightarrow R[2]$
- $S[1] \leftrightarrow S[2]$
0. Satisfying UIDs and UFDs in arity-two

**UIDs:**
- $R[2] \subseteq S[1]$
- $S[2] \subseteq R[1]$

**UFDs:**
- $R[1] \leftrightarrow R[2]$
- $S[1] \leftrightarrow S[2]$

Diagram:

```
          R
         /  \
        /    \
       a    b
          \
           \
        c --d-- e
          \
           \
        S
```
0. Satisfying UIDs and UFDs in arity-two

**UIDs:**
- $R[2] \sqsubseteq S[1]$
- $S[2] \sqsubseteq R[1]$

**UFDs:**
- $R[1] \leftrightarrow R[2]$
- $S[1] \leftrightarrow S[2]$

---

Diagram:
- Nodes: $R$, $S$, $a$, $b$, $c$, $d$, $e$
- Edges: Arrows indicate relationships between nodes.
2. Adding acyclic queries

UIDs:
R[2] \subseteq S[1]
S[2] \subseteq R[1]
and UFDs...
2. Adding acyclic queries

UIDs:

\[
S[2] \subseteq R[1]
\]

and UFDs...
2. Adding acyclic queries

query:

UIDs:


and UFDs...
2. Adding acyclic queries

query:

UIDs:


and UFDs…
2. Adding acyclic queries

query:

UIDs:


and UFDs...
2. Adding acyclic queries

query:

UIDs:

and UFDs...
Table of contents

1. Introduction
2. Context and result
3. Proof ideas
4. Conclusion
Summary

- **Finite** open-world query answering with:
  - relational instance
  - arbitrary arity signature
  - UIDs and FDs
  - conjunctive query
## Summary

- **Finite** open-world query answering with:
  - relational instance
  - arbitrary arity signature
  - UIDs and FDs
  - conjunctive query

→ We have **finite controllability up to finite closure**
Summary

- **Finite** open-world query answering with:
  - relational instance
  - arbitrary arity signature
  -UIDs and FDs
  - conjunctive query

→ We have **finite controllability up to finite closure**

- **Future work:**
  - Simplify the proof
  - More expressive frontier-one logics [Ibáñez-García et al., 2014]
  - Finite controllability and closure for path FDs in arity-two?
Summary

- **Finite** open-world query answering with:
  - relational instance
  - arbitrary arity signature
  - UIDs and FDs
  - conjunctive query

→ We have **finite controllability up to finite closure**

- **Future work:**
  - Simplify the proof
  - More expressive frontier-one logics [Ibáñez-García et al., 2014]
  - Finite controllability and closure for path FDs in arity-two?

Thanks for your attention!


*JACM*, 37(1).

Otto, M. (2002). Modal and guarded characterisation theorems over finite transition systems. In LICS.

1. Supporting arbitrary arity

When no UFDs:
reuse elements from suitable positions

Must saturate initially to make sure such elements exist

With UFDs:
bijective maps within $\longleftrightarrow$-classes
1. Supporting arbitrary arity

When no UFDs:
reuse elements from suitable positions
Must saturate initially to make sure such elements exist

With UFDs:
bijective maps within ⇄-classes
1. Supporting arbitrary arity

**UIDs:**

\[
R[2] \subseteq S[1]
\]

\[
S[2] \subseteq R[1]
\]

**UFDs:**

\[
R[1] \leftrightarrow R[2]
\]

\[
S[1] \leftrightarrow S[2]
\]
1. Supporting arbitrary arity

**UIDs:**
- $R[2] \subseteq S[1]
- $S[2] \subseteq R[1]

**UFDs:**
- $R[1] \leftrightarrow R[2]
- $S[1] \leftrightarrow S[2]

When no UFDs:
- Reuse elements from suitable positions
- Must saturate initially to make sure such elements exist

With UFDs:
- Bijective maps within $\leftrightarrow$-classes

$4/7$
1. Supporting arbitrary arity

**UIDs:**
- $R[2] \subseteq S[1]$
- $S[2] \subseteq R[1]$

**UFDs:**
- $R[1] \leftrightarrow R[2]$
- $S[1] \leftrightarrow S[2]$

When no UFDs:
- Reuse elements from suitable positions
- Must saturate initially to make sure such elements exist

With UFDs:
- Bijective maps within $\longleftrightarrow$-classes

\[4/7\]
1. Supporting arbitrary arity

**UIDs:**
- \( S[2] \subseteq R[1] \)

**UFDs:**
- \( R[1] \leftrightarrow R[2] \)
- \( S[1] \leftrightarrow S[2] \)

When no UFDs:
- Reuse elements from suitable positions
- Must saturate initially to make sure such elements exist

With UFDs:
- Bijective maps within \( \leftrightarrow \)-classes

\( 4/7 \)
1. Supporting arbitrary arity

**UIDs:**
- $R[2] \subseteq S[1]$
- $S[2] \subseteq R[1]$

**UFDs:**
- $R[1] \leftrightarrow R[2]$
- $S[1] \leftrightarrow S[2]$

- When no UFDs: **reuse** elements from suitable positions
1. Supporting arbitrary arity

**UIDs:**

- $R[2] \subseteq S[1]$
- $S[2] \subseteq R[1]$

**UFDs:**

- $R[1] \leftrightarrow R[2]$
- $S[1] \leftrightarrow S[2]$

When no UFDs: reuse elements from suitable positions

→ Must saturate initially to make sure such elements exist
1. Supporting arbitrary arity

**UIDs:**
- $R[2] \subseteq S[1]$
- $S[2] \subseteq R[1]$

**UFDs:**
- $R[1] \leftrightarrow R[2]$
- $S[1] \leftrightarrow S[2]$

---

- When no UFDs: **reuse** elements from suitable positions
  - Must **saturate** initially to make sure such elements exist
- With UFDs: **bijective** maps within $\leftrightarrow$-classes
3. From reversible to arbitrary UIDs and UFDs

```
R
(⊆ ⊆ ⊆ ⊆ ⊆)

S
(⊆)

T
U
(⊆ ⊆ ⊆)
```

UIDs can be decomposed in reversible subsets. The subsets can be solved independently. Relies on the finite implication construction [Cosmadakis et al., 1990].
3. From reversible to arbitrary UIDs and UFDs

UIDs can be decomposed in reversible subsets. The subsets can be solved independently. Relies on the finite implication construction [Cosmadakis et al., 1990].

\[
\begin{align*}
\equiv \quad & \equiv \quad \equiv \\
R & \quad S & \quad T \\
\equiv & \quad \equiv & \quad \equiv
\end{align*}
\]
3. From reversible to arbitrary UIDs and UFDs

UIDs can be decomposed in reversible subsets. The subsets can be solved independently. Relies on the finite implication construction [Cosmadakis et al., 1990].
3. From reversible to arbitrary UIDs and UFDs

UIDs can be decomposed in reversible subsets. The subsets can be solved independently. Relies on the finite implication construction [Cosmadakis et al., 1990].
3. From reversible to arbitrary UIDs and UFDs

UIDs can be decomposed in reversible subsets. The subsets can be solved independently. Relies on the finite implication construction [Cosmadakis et al., 1990].
3. From reversible to arbitrary UIDs and UFDs

- UIDs can be decomposed in reversible subsets
- The subsets can be solved independently
- Relies on the finite implication construction
  [Cosmadakis et al., 1990]
4. From UFDs to FDs

- Create initially many reusable elements
- Reuse them in new combinations

- Dense interpretations lemma: linearly many elements give super-linearly many combinations
4. From UFDs to FDs

UID:

4. From UFDs to FDs

**UID:**

**FD:**
\[ R[1,3] \rightarrow R[2] \]
4. From UFDs to FDs

UID:

FD:
\[ R[1,3] \rightarrow R[2] \]
4. From UFDs to FDs


**UID:**
$$\text{R}[1,3] \rightarrow \text{R}[2]$$

**FD:**
$$R[1,3] \rightarrow R[2]$$
4. From UFDs to FDs

**UID:**

**FD:**
\[ R[1,3] \rightarrow R[2] \]
4. From UFDs to FDs

UID:

FD:
$R[1,3] \rightarrow R[2]$
4. From UFDs to FDs

- Create initially many **reusable elements**
- Reuse them in **new combinations**

**UID:**


**FD:**

\[ R[1,3] \rightarrow R[2] \]
4. From UFDs to FDs

- Create initially many reusable elements
- Reuse them in new combinations
- Dense interpretations lemma:
  linearly many elements give super-linearly many combinations

**UID:**

**FD:**
\[ R[1,3] \rightarrow R[2] \]
5. From acyclic queries to arbitrary queries
5. From acyclic queries to arbitrary queries

UID:

\[ S[2] \subseteq R[1] \]

5. From acyclic queries to arbitrary queries

UID:

\[ S[2] \subseteq R[1] \]
5. From acyclic queries to arbitrary queries

query:

UID:

\[ S[2] \subseteq R[1] \]

From acyclic queries to arbitrary queries

UID:

\[ S[2] \subseteq R[1] \]

5. From acyclic queries to arbitrary queries

\[ S[2] \subseteq R[1] \]


**UID:**

- Red arrows: Direct dependencies
- Blue arrows: Inferred dependencies
5. From acyclic queries to arbitrary queries

UID:

\[ S[2] \subseteq R[1] \]

5. From acyclic queries to arbitrary queries

UID:

- $S[2] \subseteq R[1]$

Product with a group of high girth [Otto, 2002]
Must be tweaked to avoid violating FDs