

The Fine-Grained Complexity of Evaluating Database Queries

Nofar CARMELI

The Fine-Grained Complexity of Evaluating Queries

- To use data: store & **query**

The Fine-Grained Complexity of Evaluating Queries

- To use data: store & **query**
- Big data requires extremely efficient algorithms
 - **Fine-grained** complexity: 'polynomial' is not enough

The Fine-Grained Complexity of Evaluating Queries

- To use data: store & **query**
- Big data requires extremely efficient algorithms
 - **Fine-grained** complexity: 'polynomial' is not enough
- What is the **most efficient way** of answering a database query?

My Past Work

- Query Decompositions [PODS 17, DAM 20]
- Constant delay enumeration:
 - DBs with constraints [ICDT 18, TOCS 19]
 - Unions of CQs [PODS 19, TODS 21]
 - Random order [PODS 20]
 - Ranked access [PODS 21, PODS 23]
- Repairing noisy DBs [ICDT 21]
- Representing probabilistic DBs [PODS 21]

My Past Work

- Query Decompositions [PODS 17, DAM 20]
- Constant delay enumeration:
 - DBs with constraints [ICDT 18, TOCS 19]
 - Unions of CQs [PODS 19, TODS 21]
 - Random order [PODS 20]
 - Ranked access [PODS 21, **PODS 23**]
- Repairing noisy DBs [ICDT 21]
- Representing probabilistic DBs [PODS 21]

My Past Work

- Query Decompositions [PODS 17, DAM 20]
- Constant delay enumeration:
 - DBs with constraints [ICDT 18, TOCS 19]
 - **Unions of CQs** [PODS 19, TODS 21]
 - Random order [PODS 20]
 - Ranked access [PODS 21, PODS 23]
- Repairing noisy DBs [ICDT 21]
- Representing probabilistic DBs [PODS 21]

Enumeration Complexity of UCQs

- Goal
- Overview
- Explanations
 - Easy \cup Hard
 - Why isn't it always hard?
 - When is it easy?
 - Hard \cup Hard
 - Sometimes it is easy

Enumeration Complexity of UCQs

- **Goal**
- Overview
- Explanations
 - Easy \cup Hard
 - Why isn't it always hard?
 - When is it easy?
 - Hard \cup Hard
 - Sometimes it is easy

Goal

Which Unions of Conjunctive Queries can be answered with optimal time guarantees?

Studied Queries

tutorials:

Person	Title
Alan Fekete	Making Consistency...
Suresh Venkatasu...	Algorithmic Fairness...

schedule:

Title	Day
Making Consistency...	Tue
Algorithmic Fairness...	Wed
Regularizing Conjunct...	Mon

Studied Queries

tutorials:

Person	Title
Alan Fekete	Making Consistency...
Suresh Venkatasu...	Algorithmic Fairness...

schedule:

Title	Day
Making Consistency...	Tue
Algorithmic Fairness...	Wed
Regularizing Conjunct...	Mon

Conjunctive Query:
Join + Project

Studied Queries

tutorials:

Person	Title
Alan Fekete	Making Consistency...
Suresh Venkatasu...	Algorithmic Fairness...

schedule:

Title	Day
Making Consistency...	Tue
Algorithmic Fairness...	Wed
Regularizing Conjunct...	Mon

Conjunctive Query:
Join + Project

$Q_1(\text{Person}, \text{Day}) \leftarrow \text{tutorials}(\text{Person}, \text{Title}), \text{schedule}(\text{Title}, \text{Day})$

Studied Queries

tutorials:

Person	Title
Alan Fekete	Making Consistency...
Suresh Venkatasu...	Algorithmic Fairness...

schedule:

Title	Day
Making Consistency...	Tue
Algorithmic Fairness...	Wed
Regularizing Conjunct...	Mon

Person	Day
Alan Fekete	Tue
Suresh Venkatasu...	Wed

Conjunctive Query:
Join + Project

$Q_1(\text{Person}, \text{Day}) \leftarrow \text{tutorials}(\text{Person}, \text{Title}), \text{schedule}(\text{Title}, \text{Day})$

Studied Queries

tutorials:

Person	Title
Alan Fekete	Making Consistency...
Suresh Venkatasu...	Algorithmic Fairness...

schedule:

Title	Day
Making Consistency...	Tue
Algorithmic Fairness...	Wed
Regularizing Conjunct...	Mon

research talks:

Person	Title
Pablo Barceló	Regularizing Conjunct...
Peter Lindner	Probabilistic Database...
Muhammad Tibi	Query Evaluation in...

Person	Day
Alan Fekete	Tue
Suresh Venkatasu...	Wed

Conjunctive Query:
Join + Project

$Q_1(\text{Person}, \text{Day}) \leftarrow \text{tutorials}(\text{Person}, \text{Title}), \text{schedule}(\text{Title}, \text{Day})$

Studied Queries

tutorials:

Person	Title
Alan Fekete	Making Consistency...
Suresh Venkatasu...	Algorithmic Fairness...

schedule:

Title	Day
Making Consistency...	Tue
Algorithmic Fairness...	Wed
Regularizing Conjunct...	Mon

research talks:

Person	Title
Pablo Barceló	Regularizing Conjunct...
Peter Lindner	Probabilistic Database...
Muhammad Tibi	Query Evaluation in...

Person	Day
Alan Fekete	Tue
Suresh Venkatasu...	Wed

Conjunctive Query:
Join + Project

$Q_1(\text{Person}, \text{Day}) \leftarrow \text{tutorials}(\text{Person}, \text{Title}), \text{schedule}(\text{Title}, \text{Day})$

$Q_2(\text{Person}, \text{Day}) \leftarrow \text{research}(\text{Person}, \text{Title}), \text{schedule}(\text{Title}, \text{Day})$

Studied Queries

tutorials:

Person	Title
Alan Fekete	Making Consistency...
Suresh Venkatasu...	Algorithmic Fairness...

schedule:

Title	Day
Making Consistency...	Tue
Algorithmic Fairness...	Wed
Regularizing Conjunct...	Mon

research talks:

Person	Title
Pablo Barceló	Regularizing Conjunct...
Peter Lindner	Probabilistic Database...
Muhammad Tibi	Query Evaluation in...

Conjunctive Query:
Join + Project

Person	Day
Alan Fekete	Tue
Suresh Venkatasu...	Wed
Person	Day
Pablo Barceló	Mon
Peter Lindner	Mon
Muhammad Tibi	Mon

$Q_1(\text{Person}, \text{Day}) \leftarrow \text{tutorials}(\text{Person}, \text{Title}), \text{schedule}(\text{Title}, \text{Day})$

$Q_2(\text{Person}, \text{Day}) \leftarrow \text{research}(\text{Person}, \text{Title}), \text{schedule}(\text{Title}, \text{Day})$

Studied Queries

tutorials:

Person	Title
Alan Fekete	Making Consistency...
Suresh Venkatasu...	Algorithmic Fairness...

schedule:

Title	Day
Making Consistency...	Tue
Algorithmic Fairness...	Wed
Regularizing Conjunct...	Mon

research talks:

Person	Title
Pablo Barceló	Regularizing Conjunct...
Peter Lindner	Probabilistic Database...
Muhammad Tibi	Query Evaluation in...

Union of
Conjunctive Query:
Join + Project + Union

Person	Day
Alan Fekete	Tue
Suresh Venkatasu...	Wed
Person	Day
Pablo Barceló	Mon
Peter Lindner	Mon
Muhammad Tibi	Mon

$Q_1(\text{Person}, \text{Day}) \leftarrow \text{tutorials}(\text{Person}, \text{Title}), \text{schedule}(\text{Title}, \text{Day})$

$Q_2(\text{Person}, \text{Day}) \leftarrow \text{research}(\text{Person}, \text{Title}), \text{schedule}(\text{Title}, \text{Day})$

Studied Queries

tutorials:

Person	Title
Alan Fekete	Making Consistency...
Suresh Venkatasu...	Algorithmic Fairness...

schedule:

Title	Day
Making Consistency...	Tue
Algorithmic Fairness...	Wed
Regularizing Conjunct...	Mon

research talks:

Person	Title
Pablo Barceló	Regularizing Conjunct...
Peter Lindner	Probabilistic Database...
Muhammad Tibi	Query Evaluation in...

Union of
Conjunctive Query:
Join + Project + Union

Person	Day
Alan Fekete	Tue
Suresh Venkatasu...	Wed
Person	Day
Pablo Barceló	Mon
Peter Lindner	Mon
Muhammad Tibi	Mon

$Q_1(\text{Person}, \text{Day}) \leftarrow \text{tutorials}(\text{Person}, \text{Title}), \text{schedule}(\text{Title}, \text{Day})$

$Q_2(\text{Person}, \text{Day}) \leftarrow \text{research}(\text{Person}, \text{Title}), \text{schedule}(\text{Title}, \text{Day})$

$Q_3 = Q_1 \cup Q_2$

Studied Queries

tutorials:

Person	Title
Alan Fekete	Making Consistency...
Suresh Venkatasu...	Algorithmic Fairness...

schedule:

Title	Day
Making Consistency...	Tue
Algorithmic Fairness...	Wed
Regularizing Conjunct...	Mon

research talks:

Person	Title
Pablo Barceló	Regularizing Conjunct...
Peter Lindner	Probabilistic Database...
Muhammad Tibi	Query Evaluation in...

Union of
Conjunctive Query:
Join + Project + Union

Person	Day
Alan Fekete	Tue
Suresh Venkatasu...	Wed
Pablo Barceló	Mon
Peter Lindner	Mon
Muhammad Tibi	Mon

$Q_1(\text{Person}, \text{Day}) \leftarrow \text{tutorials}(\text{Person}, \text{Title}), \text{schedule}(\text{Title}, \text{Day})$

$Q_2(\text{Person}, \text{Day}) \leftarrow \text{research}(\text{Person}, \text{Title}), \text{schedule}(\text{Title}, \text{Day})$

$Q_3 = Q_1 \cup Q_2$

Complexity of Queries

- Query = problem

Complexity of Queries

- Query = problem
- Time complexity, data complexity, RAM model

Complexity of Queries

- Query = problem
- Time complexity, data complexity, RAM model
- Achievable time bounds:

Complexity of Queries

- Query = problem
- Time complexity, data complexity, RAM model
- Achievable time bounds:
 - Need to print every answer ($|OUT| \gg |IN|$)

Complexity of Queries

- Query = problem
- Time complexity, data complexity, RAM model
- Achievable time bounds:
 - Need to print every answer ($|OUT| \gg |IN|$)
 - Need to read the input before the first answer

Complexity of Queries

- Query = problem
- Time complexity, data complexity, RAM model
- Achievable time bounds:
 - Need to print every answer ($|OUT| \gg |IN|$)
 - Need to read the input before the first answer



Complexity of Queries

- Query = problem
- Time complexity, data complexity, RAM model
- Achievable time bounds:
 - Need to print every answer ($|OUT| \gg |IN|$)
 - Need to read the input before the first answer
- **DelayC_{lin}**: solvable in linear preprocessing and constant delay



Complexity of Queries

- Query = problem
- Time complexity, data complexity, RAM model
- Achievable time bounds:
 - Need to print every answer ($|OUT| \gg |IN|$)
 - Need to read the input before the first answer
- **DelayC_{lin}**: solvable in linear preprocessing and constant delay

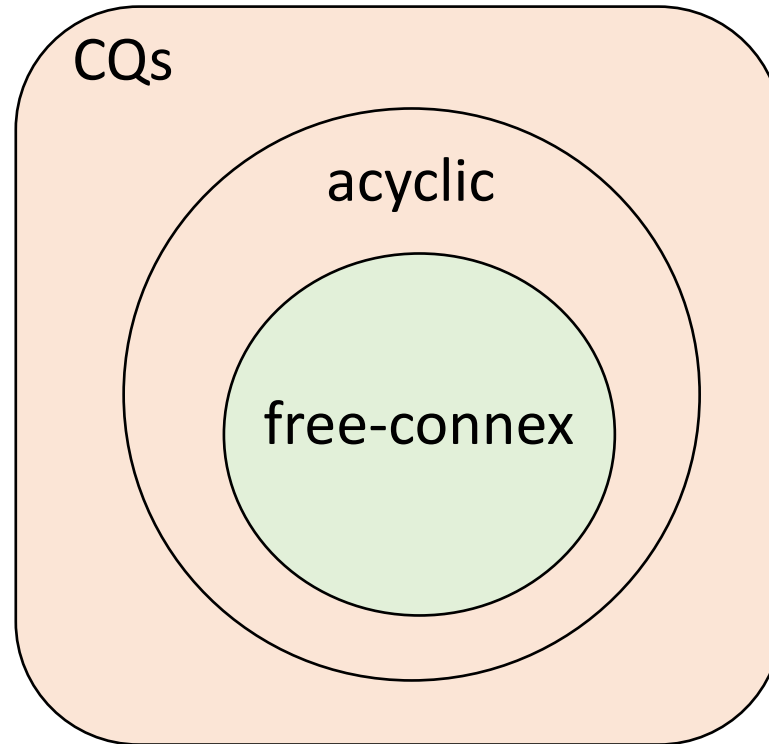


Which queries are in DelayC_{lin}?

Starting Point

[BaganDurandGrandjean CSL'2007]
[Brault-Baron 2013]

CQs: $\in DelayC_{lin} \Leftrightarrow^* \text{free-connex}$



* Hardness results assume:

- (1) no self-joins
- (2) hardness of Boolean matrix multiplication and hyperclique

Free-connex Definition

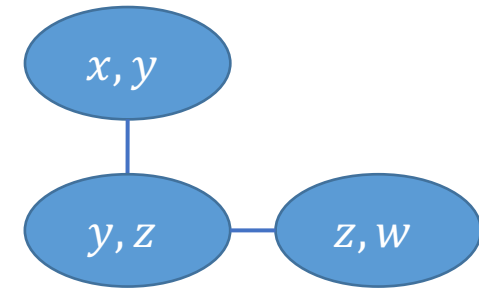
1. a node for every atom

2. tree

3. for every variable X:
the nodes containing X form a subtree

acyclic

$$Q(x, y, z) \leftarrow R_1(x, y), R_2(y, z), R_3(z, w)$$



Free-connex Definition

1. a node for every atom

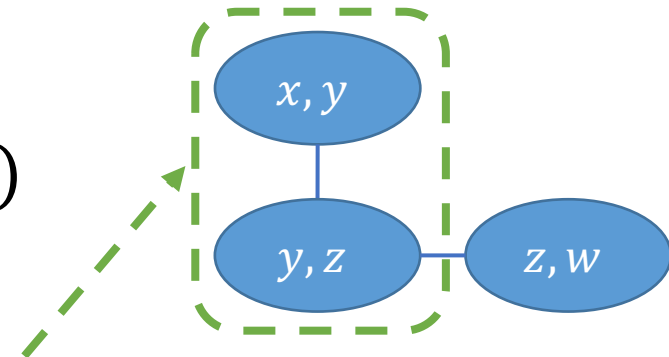
2. tree

3. for every variable X:
the nodes containing X form a subtree

acyclic

free – connex

$$Q(x, y, z) \leftarrow R_1(x, y), R_2(y, z), R_3(z, w)$$



4. a subtree with exactly the free variables

Free-connex Definition

1. a node for every atom
possibly also subsets

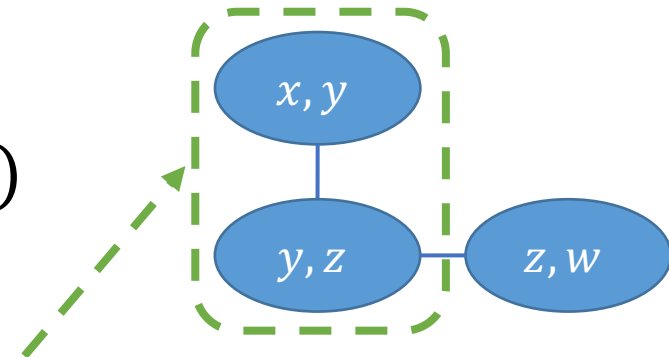
2. tree

3. for every variable X:
the nodes containing X form a subtree

acyclic

free – connex

$$Q(x, y, z) \leftarrow R_1(x, y), R_2(y, z), R_3(z, w)$$

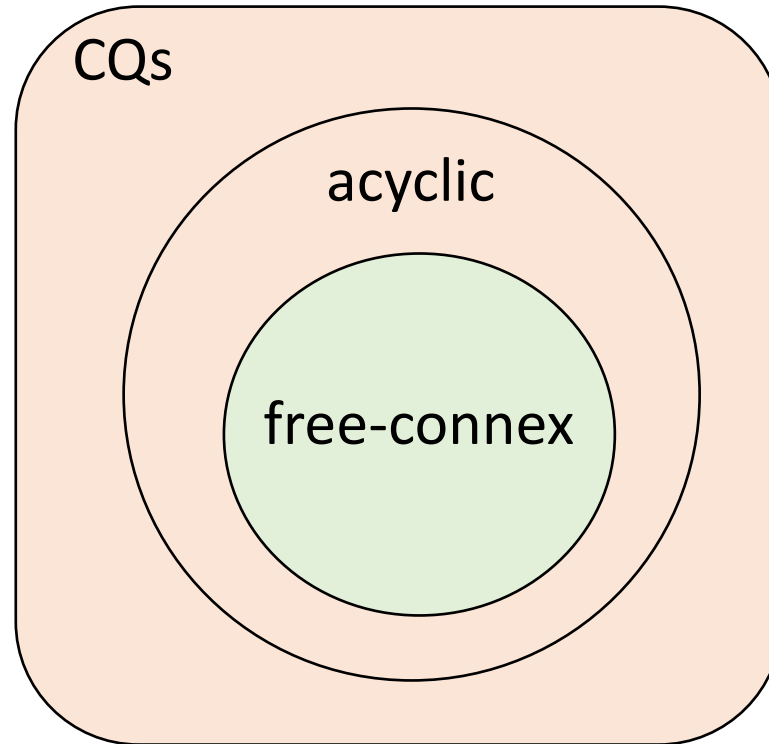


4. a subtree with exactly the free variables

Starting Point

[BaganDurandGrandjean CSL'2007]
[Brault-Baron 2013]

CQs: $\in DelayC_{lin} \Leftrightarrow^* \text{free-connex}$



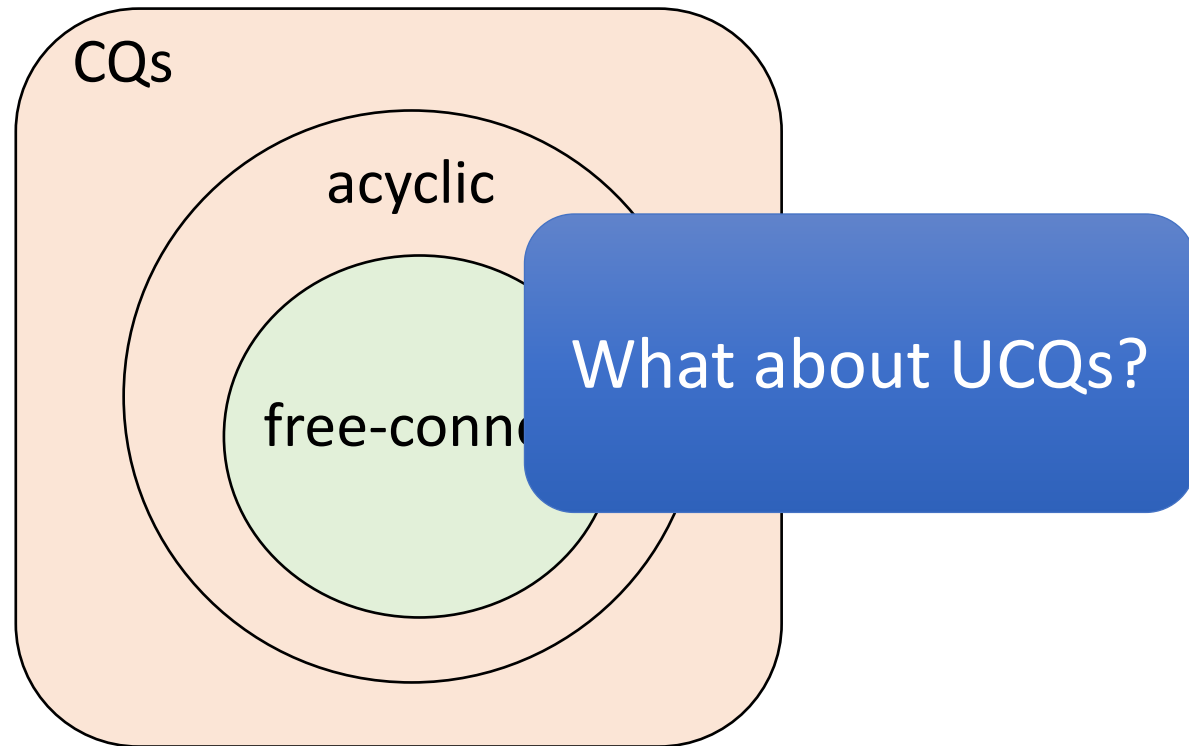
* Hardness results assume:

- (1) no self-joins
- (2) hardness of Boolean matrix multiplication and hyperclique

Starting Point

[BaganDurandGrandjean CSL'2007]
[Brault-Baron 2013]

CQs: $\in DelayC_{lin} \Leftrightarrow^* \text{free-connex}$



* Hardness results assume:

- (1) no self-joins
- (2) hardness of Boolean matrix multiplication and hyperclique

Enumeration Complexity of UCQs

- Goal
- **Overview**
- Explanations
 - Easy \cup Hard
 - Why isn't it always hard?
 - When is it easy?
 - Hard \cup Hard
 - Sometimes it is easy

Cases for UCQs

All CQs are Easy

Some Easy, Some Hard

All CQs are Hard

Cases for UCQs

All CQs are Easy

Some Easy, Some Hard

All CQs are Hard

Cases for UCQs

All CQs are Easy

always easy

Some Easy, Some Hard

All CQs are Hard

Cases for UCQs

All CQs are Easy

always easy

Some Easy, Some Hard

All CQs are Hard

Cases for UCQs

All CQs are Easy

always easy

Some Easy, Some Hard

sometimes hard

All CQs are Hard

Cases for UCQs

All CQs are Easy

always easy

Some Easy, Some Hard

sometimes hard

All CQs are Hard

unions **with** a hard CQ
can be equivalent to
unions **without** a hard CQs

Cases for UCQs

All CQs are Easy

always easy

Some Easy, Some Hard

sometimes hard

All CQs are Hard

unions **with** a hard CQ
can be equivalent to
unions **without** a hard CQs

$$Q_1(x, y) \leftarrow R_1(x, y), R_2(y, z), R_3(z, x)$$

$$Q_2(x, y) \leftarrow R_1(x, y), R_2(y, z)$$

Cases for UCQs

All CQs are Easy

always easy

Some Easy, Some Hard

sometimes hard

All CQs are Hard

unions **with** a hard CQ
can be equivalent to
unions **without** a hard CQs

$Q_1(x, y) \leftarrow R_1(x, y), R_2(y, z), R_3(z, x)$

non free – connex

$Q_2(x, y) \leftarrow R_1(x, y), R_2(y, z)$

free – connex

Cases for UCQs

All CQs are Easy

always easy

Some Easy, Some Hard

sometimes hard

All CQs are Hard

unions **with** a hard CQ
can be equivalent to
unions **without** a hard CQs

$Q_1(x, y) \leftarrow R_1(x, y), R_2(y, z), R_3(z, x)$

non free – connex

$Q_2(x, y) \leftarrow R_1(x, y), R_2(y, z)$

free – connex

$Q_1 \subseteq Q_2$

Cases for UCQs

All CQs are Easy

always easy

Some Easy, Some Hard

sometimes hard

All CQs are Hard

unions **with** a hard CQ
can be equivalent to
unions **without** a hard CQs

$Q_1(x, y) \leftarrow R_1(x, y), R_2(y, z), R_3(z, x)$

non free – connex

$Q_2(x, y) \leftarrow R_1(x, y), R_2(y, z)$

free – connex

$$Q_1 \subseteq Q_2 \quad \Rightarrow \quad Q_1 \cup Q_2 = Q_2$$

Cases for UCQs

All CQs are Easy

always easy

Some Easy, Some Hard

sometimes hard

sometimes easy

All CQs are Hard

unions **with** a hard CQ
can be equivalent to
unions **without** a hard CQs

$Q_1(x, y) \leftarrow R_1(x, y), R_2(y, z), R_3(z, x)$

non free – connex

$Q_2(x, y) \leftarrow R_1(x, y), R_2(y, z)$

free – connex

$$Q_1 \subseteq Q_2 \quad \Rightarrow \quad Q_1 \cup Q_2 = Q_2$$

Cases for UCQs

All CQs are Easy

always easy

Some Easy, Some Hard

sometimes hard

sometimes easy

All CQs are Hard

non-redundant unions?

Cases for UCQs

All CQs are Easy

always easy

Some Easy, Some Hard

sometimes hard

sometimes easy

All CQs are Hard

non-redundant unions?

Claimed [ICDT 2018]:
hard if contains a hard CQ

Cases for UCQs

All CQs are Easy

always easy

Some Easy, Some Hard

sometimes hard

sometimes easy

All CQs are Hard

some **non-redundant** unions
with a hard CQ
are **easy**

Cases for UCQs

All CQs are Easy

always easy

Some Easy, Some Hard

sometimes hard

*sometimes easy **

All CQs are Hard

* Even for non-redundant unions

some **non-redundant** unions
with a hard CQ
are **easy**

Cases for UCQs

All CQs are Easy

always easy

Some Easy, Some Hard

sometimes hard

*sometimes easy **

All CQs are Hard

* Even for non-redundant unions

some **non-redundant** unions
with a hard CQ
are **easy**

Cases for UCQs

All CQs are Easy

always easy

Some Easy, Some Hard

sometimes hard

*sometimes easy **

* Even for non-redundant unions

All CQs are Hard

If each CQ in Q is hard
and
there is no body-isomorphism $\Rightarrow Q \notin DelayC_{lin}$

Cases for UCQs

All CQs are Easy

always easy

Some Easy, Some Hard

sometimes hard

*sometimes easy **

All CQs are Hard

* Even for non-redundant unions

UCQs containing **only hard CQs**
can be easy!

Cases for UCQs



* Even for non-redundant unions

UCQs containing **only hard CQs**
can be easy!

Cases for UCQs



* Even for non-redundant unions

UCQs containing **only hard CQs**
can be easy!

Enumeration Complexity of UCQs

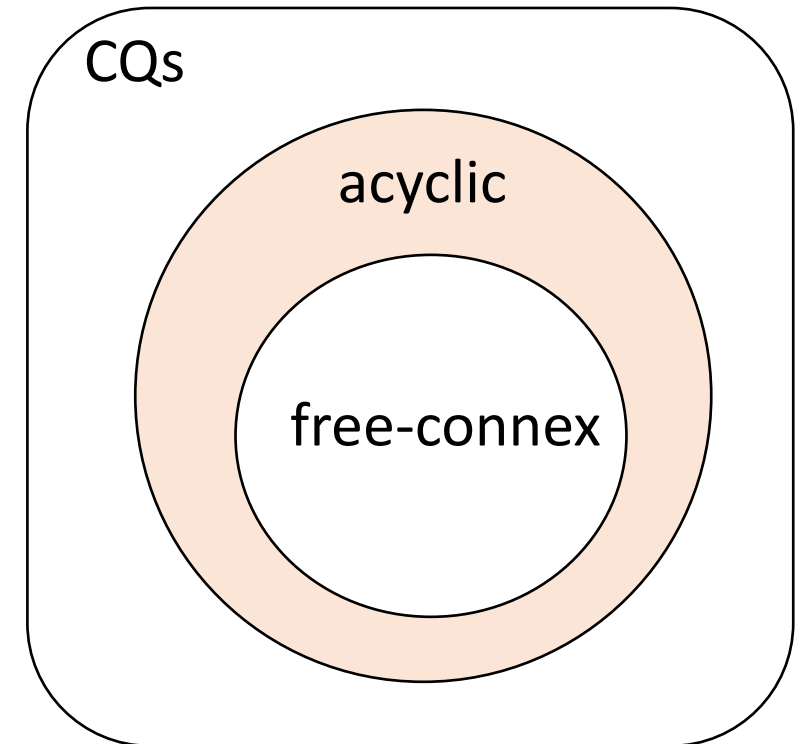
- Goal
- Overview
- Explanations
 - Easy \cup Hard
 - **Why isn't it always hard?**
 - When is it easy?
 - Hard \cup Hard
 - Sometimes it is easy

Lower Bound

[BaganDurandGrandjean CSL'2007]

Acyclic non-free-connex:

$$Q(x, z) \leftarrow R_1(x, y), R_2(y, z)$$



Lower Bound

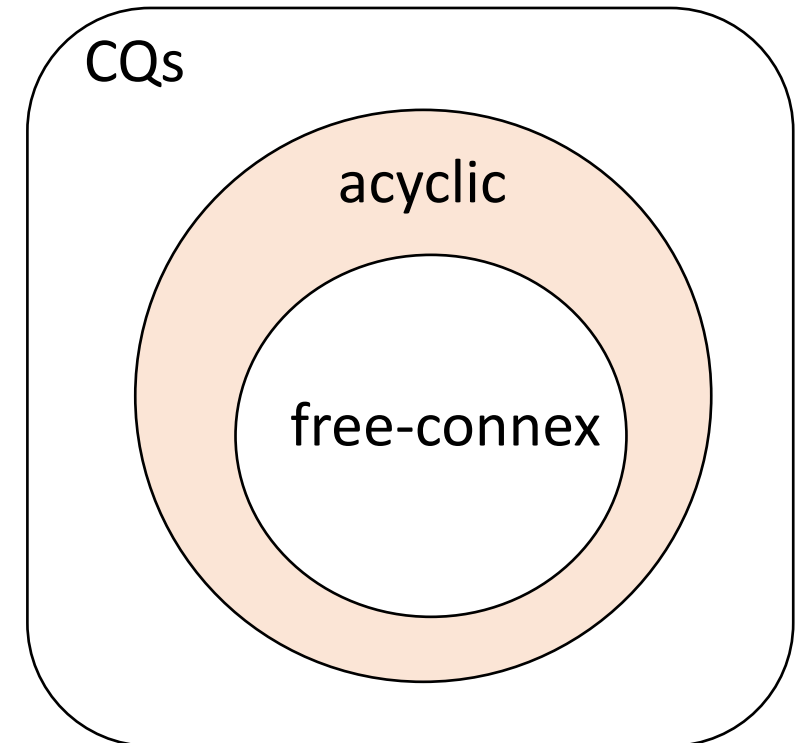
[BaganDurandGrandjean CSL'2007]

Assumption: Boolean $n \times n$ matrices cannot be multiplied in time $O(n^2)$

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} ? & ? \\ ? & ? \end{pmatrix}$$

Acyclic non-free-connex:

$$Q(x, z) \leftarrow R_1(x, y), R_2(y, z)$$



Lower Bound

[BaganDurandGrandjean CSL'2007]

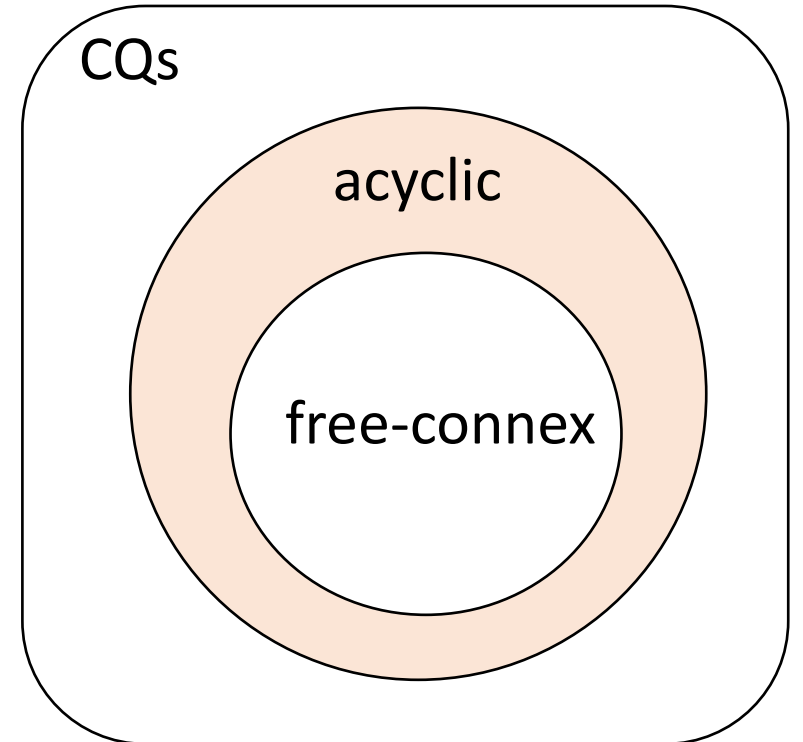
Assumption: Boolean $n \times n$ matrices cannot be multiplied in time $O(n^2)$

$$\underbrace{\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}}_A \underbrace{\begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}}_B = \begin{pmatrix} ? & ? \\ ? & ? \end{pmatrix}$$

Acyclic non-free-connex:

$$Q(x, z) \leftarrow R_1(x, y), R_2(y, z)$$

A		B	
R	C	R	C
1	1	1	2
1	2	2	2
2	2		



Lower Bound

[BaganDurandGrandjean CSL'2007]

Assumption: Boolean $n \times n$ matrices cannot be multiplied in time $O(n^2)$

$$\underset{\mathbf{A}}{\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}} \underset{\mathbf{B}}{\begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}} = \begin{pmatrix} ? & ? \\ ? & ? \end{pmatrix}$$

Acyclic non-free-connex:

$$Q(x, z) \leftarrow R_1(x, y), R_2(y, z)$$

Q

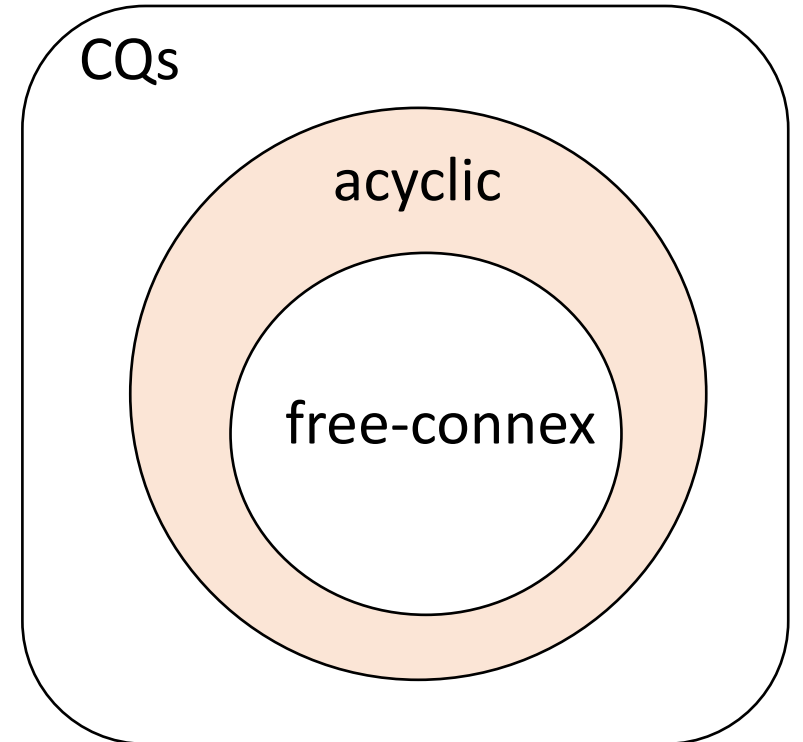
R	C
1	2
2	2

A

R	C
1	1
1	2
2	2

B

R	C
1	2
2	2



Lower Bound

[BaganDurandGrandjean CSL'2007]

Assumption: Boolean $n \times n$ matrices cannot be multiplied in time $O(n^2)$

$$\underbrace{\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}}_A \underbrace{\begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}}_B = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}$$

Acyclic non-free-connex:

$$Q(x, z) \leftarrow R_1(x, y), R_2(y, z)$$

Q

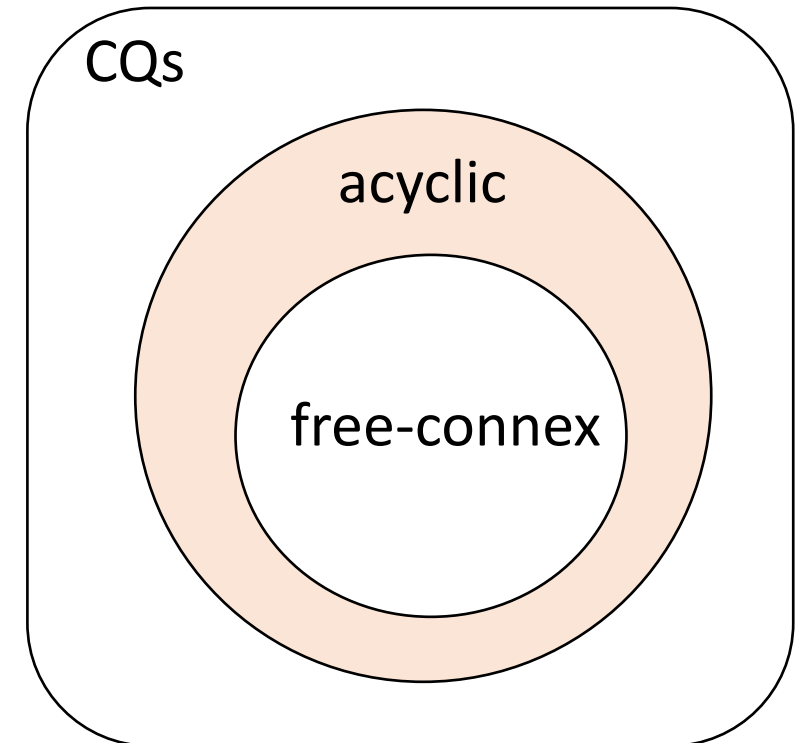
R	C
1	2
2	2

A

R	C
1	1
1	2
2	2

B

R	C
1	2
2	2



Lower Bound

[BaganDurandGrandjean CSL'2007]

Assumption: Boolean $n \times n$ matrices cannot be multiplied in time $O(n^2)$

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & \textcircled{1} \\ 0 & 1 \end{pmatrix}$$

$\textcolor{blue}{A} \qquad \textcolor{blue}{B}$

Acyclic non-free-connex:

$$Q(x, z) \leftarrow R_1(x, y), R_2(y, z)$$

$\textcolor{blue}{Q}$

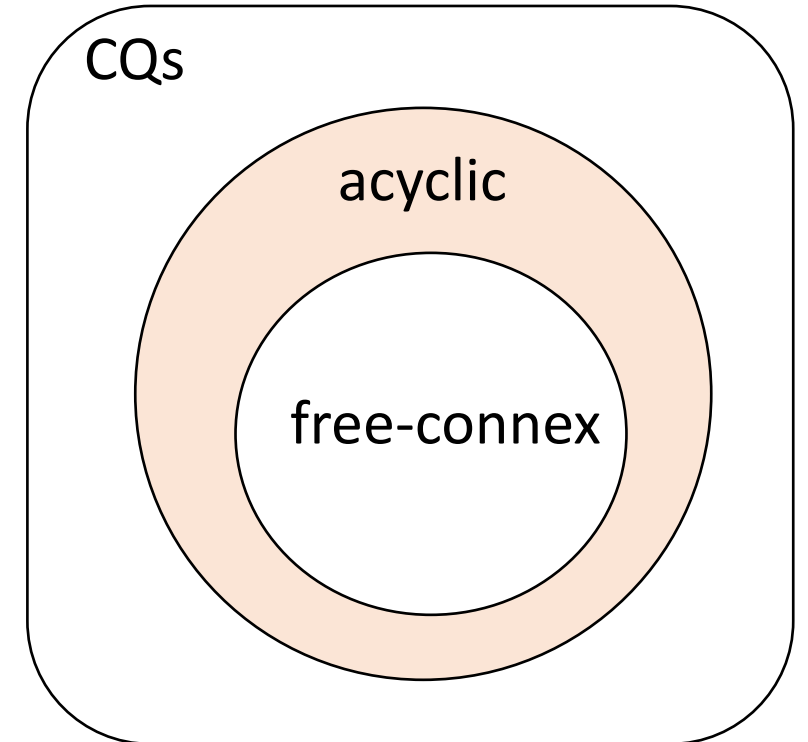
R	C
1	2
2	2

$\textcolor{blue}{A}$

R	C
1	1
1	2
2	2

$\textcolor{blue}{B}$

R	C
1	2
2	2



Lower Bound

[BaganDurandGrandjean CSL'2007]

Assumption: Boolean $n \times n$ matrices cannot be multiplied in time $O(n^2)$

$$\underbrace{\begin{pmatrix} 1 & \textcircled{1} \\ 0 & 1 \end{pmatrix}}_A \underbrace{\begin{pmatrix} 0 & 1 \\ 0 & \textcircled{1} \end{pmatrix}}_B = \begin{pmatrix} 0 & \textcircled{1} \\ 0 & 1 \end{pmatrix}$$

Acyclic non-free-connex:

$$Q(x, z) \leftarrow R_1(x, y), R_2(y, z)$$

Q

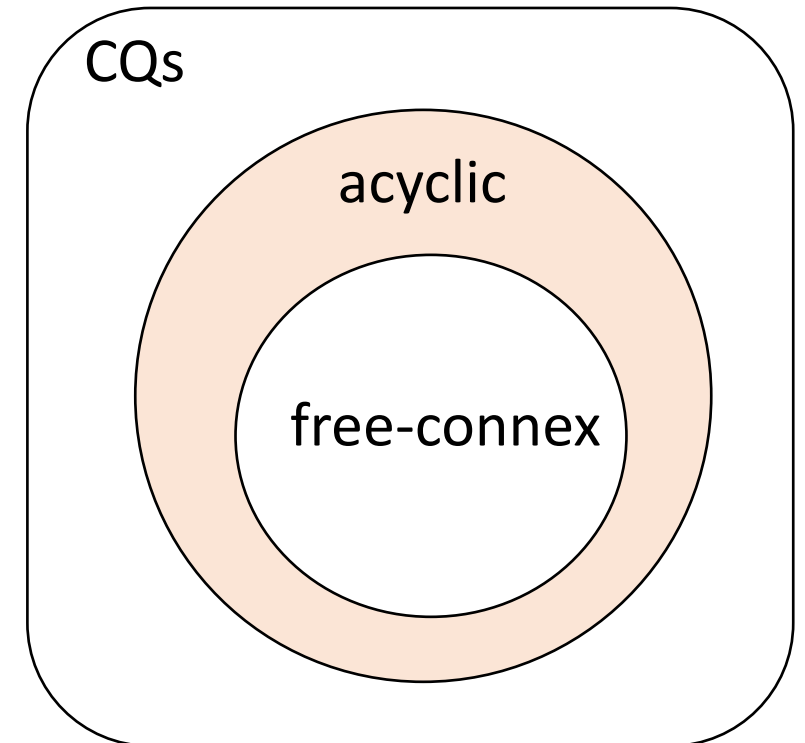
R	C
1	2
2	2

A

R	C
1	1
1	2
2	2

B

R	C
1	2
2	2



Lower Bound

[BaganDurandGrandjean CSL'2007]

Assumption: Boolean $n \times n$ matrices cannot be multiplied in time $O(n^2)$

$$\underset{\mathbf{A}}{\begin{pmatrix} 1 & \textcircled{1} \\ 0 & 1 \end{pmatrix}} \underset{\mathbf{B}}{\begin{pmatrix} 0 & 1 \\ 0 & \textcircled{1} \end{pmatrix}} = \begin{pmatrix} 0 & \textcircled{1} \\ 0 & 1 \end{pmatrix}$$

Acyclic non-free-connex:

$$Q(x, z) \leftarrow R_1(x, y), R_2(y, z)$$

$\notin \text{DelayC}_{\text{lin}}$

Q

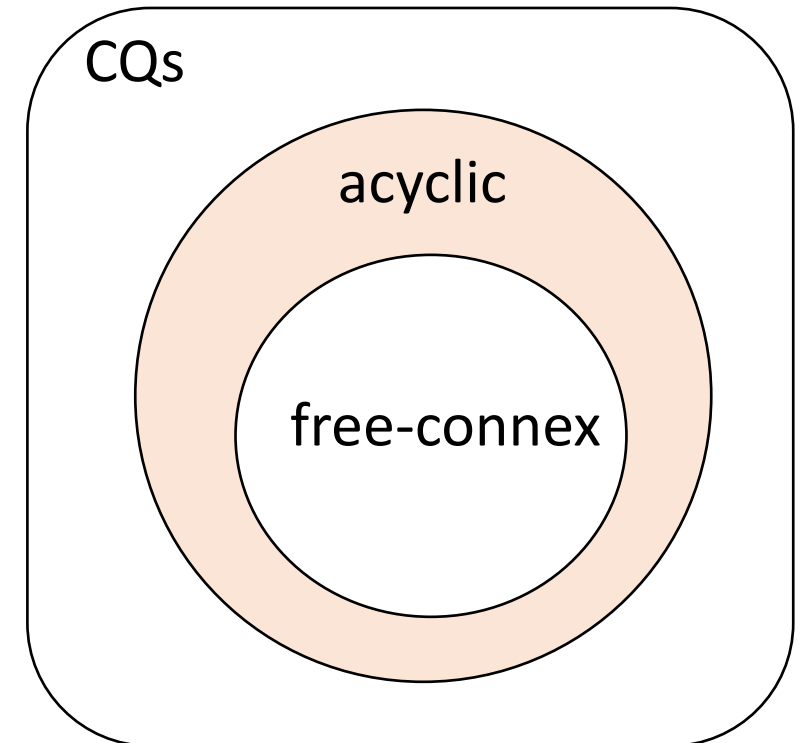
R	C
1	2
2	2

A

R	C
1	1
1	2
2	2

B

R	C
1	2
2	2



Within Unions

$$Q_1(x, z, w) \leftarrow R_1(x, y), R_2(y, z), R_3(z, w)$$

not free connex

Within Unions

$$Q_1(x, z, w) \leftarrow R_1(x, y), R_2(y, z), R_3(z, w)$$

not free connex

R_1	
1	1
1	2
2	2

R_2	
1	2
2	2

Within Unions

$$Q_1(x, z, w) \leftarrow R_1(x, y), R_2(y, z), R_3(z, w)$$

not free connex

Q_1		
1	2	\perp
2	2	\perp

R_1	
1	1
1	2
2	2

R_2	
1	2
2	2

R_3	
2	\perp

Within Unions

$$Q_1(x, z, w) \leftarrow R_1(x, y), R_2(y, z), R_3(z, w)$$
$$\cup$$

$$Q_2(x', y', z') \leftarrow R_1(x', y'), R_2(y', z')$$

not free connex

Q_1		
1	2	\perp
2	2	\perp

R_1	
1	1
1	2
2	2

R_2	
1	2
2	2

R_3	
2	\perp

Within Unions

$$Q_1(x, z, w) \leftarrow R_1(x, y), R_2(y, z), R_3(z, w)$$
$$\cup$$

$$Q_2(x', y', z') \leftarrow R_1(x', y'), R_2(y', z')$$

not free connex

Q_1		
1	2	\perp
2	2	\perp
Q_2		
1	1	2
1	2	2
2	2	2

R_1	
1	1
1	2
2	2

R_2	
1	2
2	2

R_3	
2	\perp

Within Unions

$$Q_1(x, z, w) \leftarrow R_1(x, y), R_2(y, z), R_3(z, w)$$
$$\cup$$

$$Q_2(x', y', z') \leftarrow R_1(x', y'), R_2(y', z')$$

not free connex

Q_1		
1	2	\perp
2	2	\perp
Q_2		
1	1	2
1	2	2
2	2	2

$O(n^3)$ solutions:
The computation does not
contradict the assumption

R_1	
1	1
1	2
2	2

R_2	
1	2
2	2

R_3	
2	\perp

Within Unions

$$Q_1(x, z, w) \leftarrow R_1(x, y), R_2(y, z), R_3(z, w)$$

$$\cup$$

$$Q_2(x', y', z') \leftarrow R_1(x', y'), R_2(y', z')$$

not free connex

Q_1		
1	2	\perp
2	2	\perp
Q_2		
1	1	2
1	2	2
2	2	2

$O(n^3)$ solutions:
The computation does not
contradict the assumption

R_1	
1	1
1	2
2	2

R_2	
1	2
2	2

R_3	
2	\perp

The hardness results do not hold within a union

Enumeration Complexity of UCQs

- Goal
- Overview
- Explanations
 - Easy \cup Hard
 - Why isn't it always hard?
 - **When is it easy?**
 - Hard \cup Hard
 - Sometimes it is easy

Providing Variables

$$\begin{array}{l} \text{non free – connex} \\ \text{free – connex} \end{array} \left\{ \begin{array}{l} Q_1(x, z, w) \leftarrow R_1(x, y), R_2(y, z), R_3(z, w) \\ \cup \\ Q_2(x', y', z') \leftarrow R_1(x', y'), R_2(y', z') \end{array} \right\} \in \text{DelayC}_{\text{lin}}$$

Providing Variables

$$\begin{array}{l} \text{non free – connex} \quad Q_1(x, z, w) \leftarrow \overset{\text{hard part}}{R_1(x, y), R_2(y, z), R_3(z, w)} \\ \cup \\ \text{free – connex} \quad Q_2(x', y', z') \leftarrow R_1(x', y'), R_2(y', z') \end{array} \quad \left. \vphantom{\begin{array}{l} Q_1 \\ Q_2 \end{array}} \right\} \in \text{DelayC}_{\text{lin}}$$

Providing Variables

non free – connex $Q_1(x, z, w) \leftarrow R_1(x, y), R_2(y, z), R_3(z, w)$

free – connex $Q_2(x', y', z') \leftarrow R_1(x', y'), R_2(y', z')$

\cup

Body-homomorphism

hard part

$\in \text{DelayC}_{\text{lin}}$

The diagram illustrates the relationship between two queries, Q_1 and Q_2 . $Q_1(x, z, w) \leftarrow R_1(x, y), R_2(y, z), R_3(z, w)$ is labeled 'non free – connex' in a red box. $Q_2(x', y', z') \leftarrow R_1(x', y'), R_2(y', z')$ is labeled 'free – connex' in a green box. A union symbol \cup is placed between the two queries. A purple arrow labeled 'Body-homomorphism' points from $R_1(x', y')$ in Q_2 to $R_1(x, y)$ in Q_1 . Another purple arrow points from $R_2(y', z')$ in Q_2 to $R_2(y, z)$ in Q_1 . A red label 'hard part' points to the $R_1(x, y)$ and $R_2(y, z)$ atoms in Q_1 . A green bracket on the right groups both queries and is labeled $\in \text{DelayC}_{\text{lin}}$.

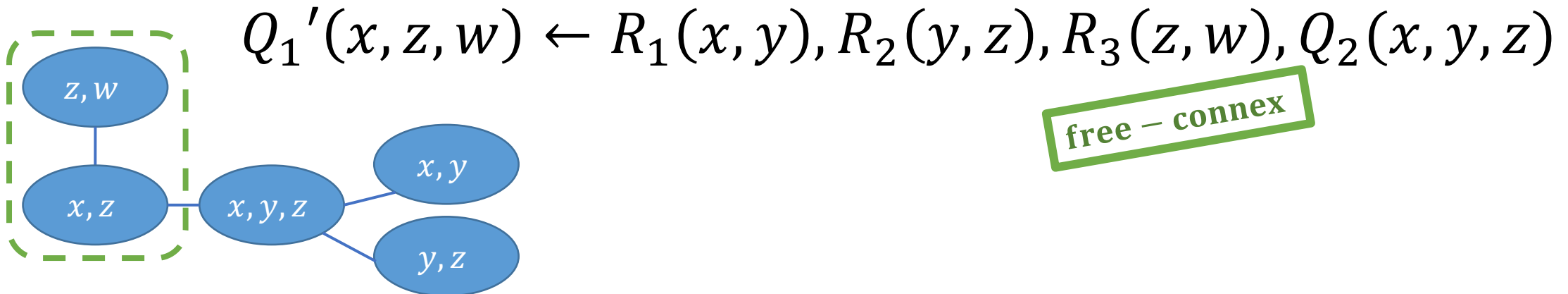
Providing Variables

$$\begin{array}{l}
 \text{non free – connex} \quad Q_1(x, z, w) \leftarrow \overbrace{R_1(x, y), R_2(y, z), R_3(z, w)}^{\text{hard part}} \\
 \text{free – connex} \quad Q_2(x', y', z') \leftarrow R_1(x', y'), R_2(y', z')
 \end{array}
 \quad \bigcup \quad
 \left. \begin{array}{l}
 \text{Body-homomorphism} \\
 \text{Body-homomorphism} \\
 \text{Body-homomorphism} \\
 \text{Body-homomorphism}
 \end{array} \right\} \in \text{DelayC}_{\text{lin}}$$

$$Q_1'(x, z, w) \leftarrow R_1(x, y), R_2(y, z), R_3(z, w), Q_2(x, y, z)$$

Providing Variables

$$\begin{array}{l}
 \text{non free – connex} \quad Q_1(x, z, w) \leftarrow \text{hard part } R_1(x, y), R_2(y, z), R_3(z, w) \\
 \text{free – connex} \quad Q_2(x', y', z') \leftarrow \text{Body-homomorphism } R_1(x', y'), R_2(y', z')
 \end{array}
 \quad \left. \vphantom{\begin{array}{l} Q_1 \\ Q_2 \end{array}} \right\} \in \text{DelayC}_{\text{lin}}$$



Enumeration Complexity of UCQs

- Goal
- Overview
- Explanations
 - Easy \cup Hard
 - Why isn't it always hard?
 - When is it easy?
 - Hard \cup Hard
 - **Sometimes it is easy**

Hard \cup Hard = Easy

- Example: CQs with **isomorphic bodies**.

$$Q_1(x, z, w, u) \leftarrow R_1(x, y), R_2(y, z), R_3(z, w), R_4(w, u)$$

$$Q_2(x, y, z, u) \leftarrow R_1(x, y), R_2(y, z), R_3(z, w), R_4(w, u)$$

Hard \cup Hard = Easy

- Example: CQs with **isomorphic bodies**.

$$\begin{aligned} Q_1(x, z, w, u) &\leftarrow \overset{\text{hard part}}{R_1(x, y), R_2(y, z)}, R_3(z, w), R_4(w, u) \\ Q_2(x, y, z, u) &\leftarrow R_1(x, y), R_2(y, z), \underset{\text{hard part}}{R_3(z, w), R_4(w, u)} \end{aligned}$$

Hard \cup Hard = Easy

- Example: CQs with **isomorphic bodies**.

$$\begin{aligned} Q_1(x, z, w, u) &\leftarrow \overset{\text{hard part}}{R_1(x, y), R_2(y, z)}, R_3(z, w), R_4(w, u) \\ Q_2(x, y, z, u) &\leftarrow R_1(x, y), R_2(y, z), \underset{\text{hard part}}{R_3(z, w), R_4(w, u)} \end{aligned}$$

	Step	Output	Side Effect
1	Solve Q_2'	$\subseteq Q_2$	Find $R_1 \bowtie R_2$
2	Solve Q_1^+	Q_1	Find $R_3 \bowtie R_4$
3	Solve Q_2^+	Q_2	

Enumeration Complexity of UCQs

- Goal
- Overview
- Explanations
 - Easy \cup Hard
 - Why isn't it always hard?
 - When is it easy?
 - Hard \cup Hard
 - Sometimes it is easy

Research Project

The Fine-Grained Complexity of Evaluating Queries

- To use data: store & **query**
- Big data requires extremely efficient algorithms
 - **Fine-grained** complexity: 'polynomial' is not enough
- What is the **most efficient way** of answering a database query?

The Fine-Grained Complexity of Evaluating Queries

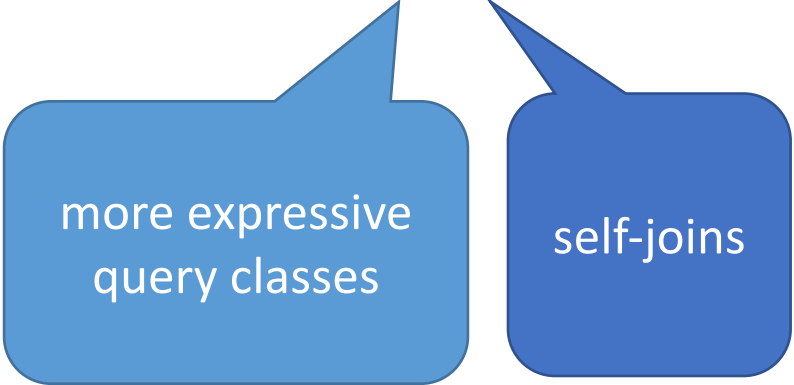
What is the most efficient way
of answering a database query?

The Fine-Grained Complexity of Evaluating Queries

What is the most efficient way
of answering a database **query**?

The Fine-Grained Complexity of Evaluating Queries

What is the most efficient way
of answering a database **query**?



more expressive
query classes

self-joins

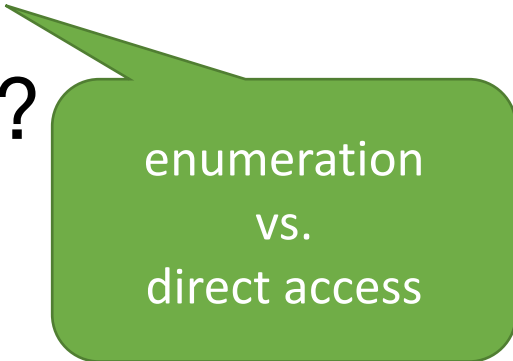
[C+, PODS'19][C+, TODS'21]

The Fine-Grained Complexity of Evaluating Queries

What is the most efficient **way**
of answering a database query?

The Fine-Grained Complexity of Evaluating Queries

What is the most efficient **way**
of answering a database query?



enumeration
vs.
direct access

[C+, PODS'20]

[C+, PODS'21]

The Fine-Grained Complexity of Evaluating Queries

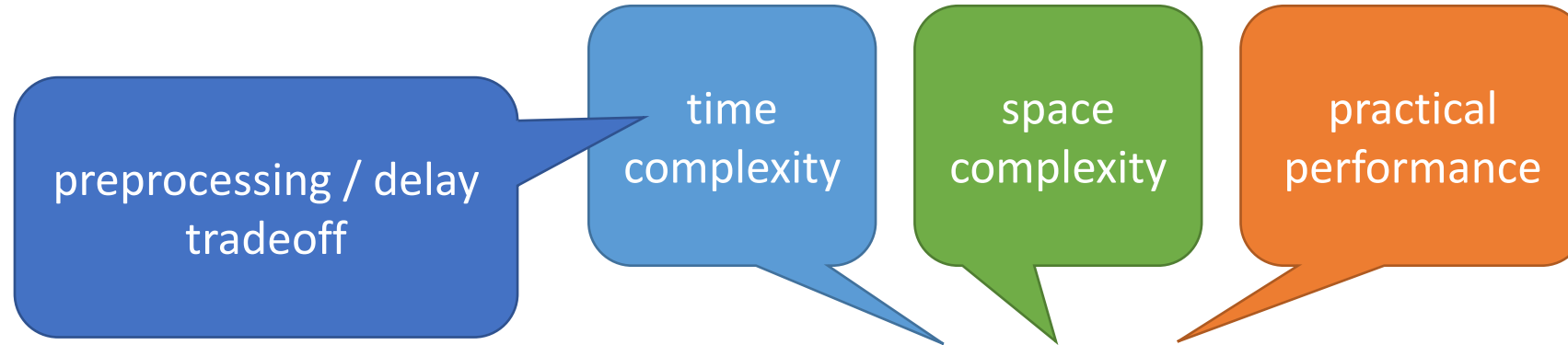
What is the most **efficient** way
of answering a database query?

The Fine-Grained Complexity of Evaluating Queries



What is the most **efficient** way
of answering a database query?

The Fine-Grained Complexity of Evaluating Queries



What is the most **efficient** way
of answering a database query?

The Fine-Grained Complexity of Evaluating Queries

What is the most efficient way
of answering a **database** query?

The Fine-Grained Complexity of Evaluating Queries

What is the most efficient way
of answering a **database** query?



[C+, ICDT'18][C+, TOCS'19]

The Fine-Grained Complexity of Evaluating Queries

**What is the most efficient way
of answering a database query?**

Integration

	IRIF	LaBRI	LIRMM
Example common interests	Graph DBs Uncertain data	DBs with constraints Ontologies	Ontologies Dynamic data
Natural collaborators	Cristina Sirangelo Amélie Gheerbrant	Diego Figueira Meghyn Bienvenu	David Carral Federico Ulliana

Thank you.