The Fine-Grained Complexity of Answering Database Queries

Nofar Carmeli

Joint work with:
Christoph Berkholz, Alessio Conte, Wolfgang Gatterbauer,
Benny Kimelfeld, Markus Kröll, Mirek Riedewald,
Nicole Schweikardt, Nikolaos Tziavelis, Shai Zeevi.

Content

1. Intro: Join Queries and Acyclicity
2. Enumeration Complexity:
    - Dependencies
    - Unions
3. Additional Tasks:
    - Random Order
    - Ranked Access
    - Results: CQs, UCQs

Content

1. **Intro: Join Queries and Acyclicity**
2. Enumeration Complexity:
   - Dependencies
   - Unions
3. Additional Tasks:
   - Random Order
   - Ranked Access
   - Results: CQs, UCQs

Example: Join

Database

Completed

StudentID	ClassNumber
319489023	234846
298473829	234846

Classes

ClassNumber	ClassName
234846	Algorithms
236342	Magicology

Join Query

Completed(StudentID,ClassNumber) ⋈ Classes(ClassNumber,ClassName)

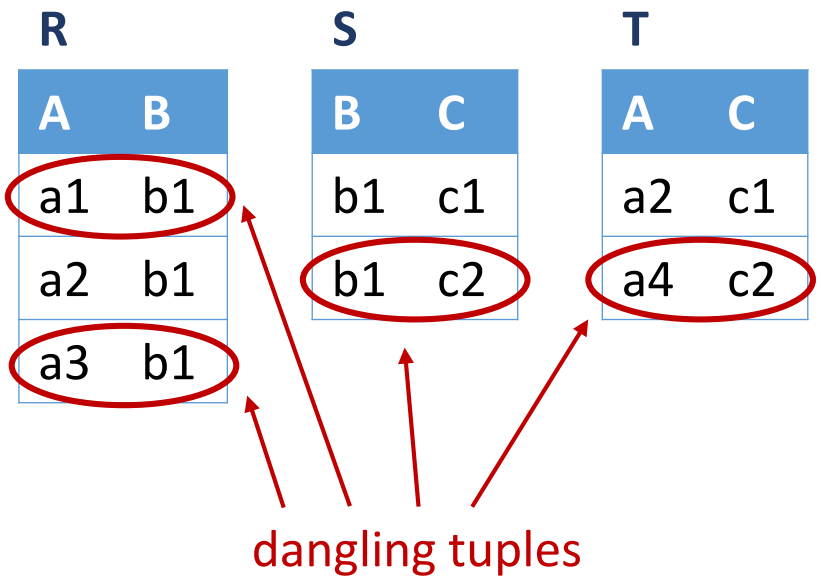
Query Result

StudentID	ClassNumber	ClassName
319489023	234846	Algorithms
298473829	234846	Algorithms

$$\{(\text{StudentID}, \text{ClassNumber}, \text{ClassName}) \mid (\text{StudentID}, \text{ClassNumber}) \in \text{Completed}, (\text{ClassNumber}, \text{ClassName}) \in \text{Classes}\}$$

Challenges

- Many answers
- Many intermediate answers



R(A,B)⋈S(B,C)

A	B	C
a1	b1	c1
a1	b1	c2
a2	b1	c1
a2	b1	c2
a3	b1	c1
a3	b1	c2

R(A,B)⋈S(B,C)⋈T(A,C)

A	B	C
a2	b1	c1

Complexity of Query Answering

- How to formulate the complexity?
  - Need to read the input before the first answer
  - Need to print every answer (|OUT| >> |IN|)
- Natural complexity measure:
  - linear preprocessing
  - Constant delay
- Goal: **when** can we reach this?

Acyclicity

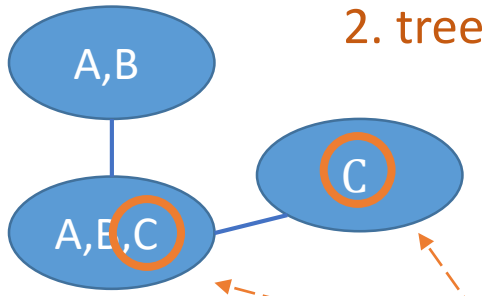
- Acyclic ⟹ solvable with linear preprocessing time and constant delay.

Join Tree:

1. a node for every relation

2. tree

Query: $R(A,B) \bowtie S(A,B,C) \bowtie T(C)$ **acyclic**

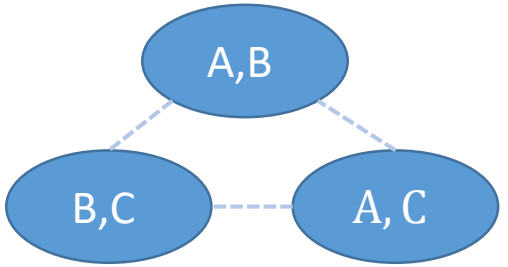


3. For every attribute X:
The nodes containing X form a subtree

- A query that has a join tree is called <u>acyclic</u>

cyclic

Query: $R(A,B) \bowtie S(B,C) \bowtie T(A,C)$

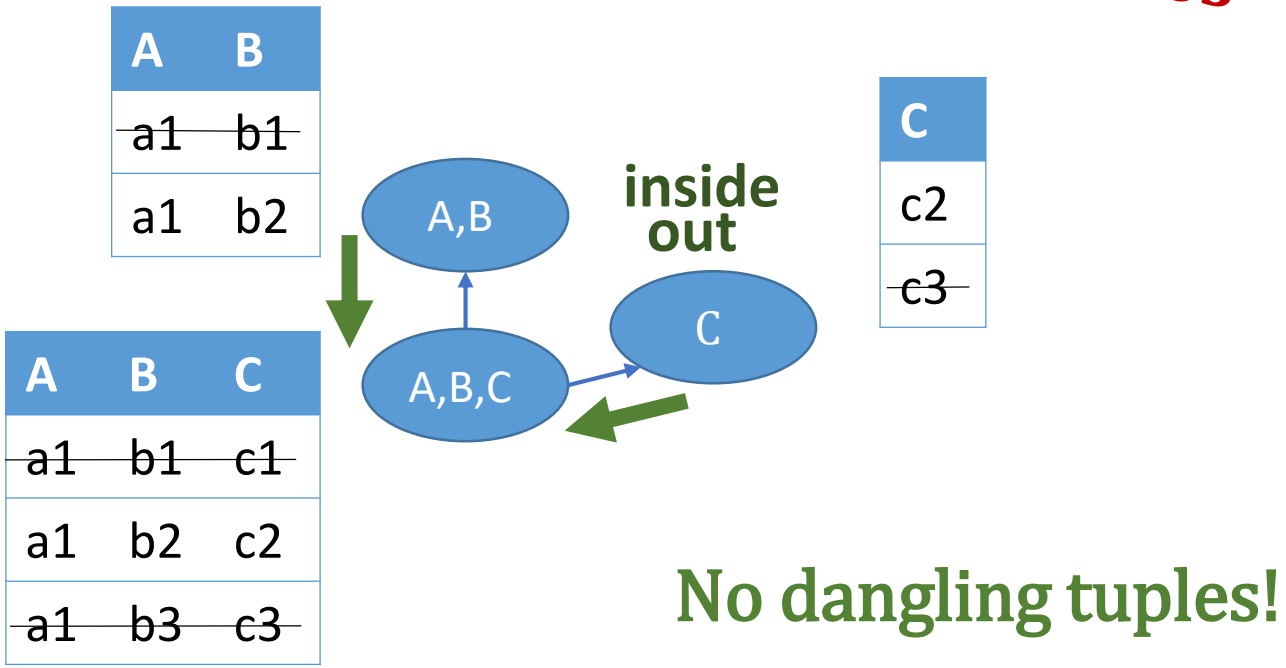


Acyclic Joins

- An efficient algorithm for acyclic joins [Yannakakis81]
  1. Find a join tree and set a root
  2. Remove dangling tuples
  3. Join

only works with join trees

1. Leaf-to-root:
   $r_{parent} \leftarrow r_{parent} \ltimes r_v$
2. Root-to-leaf:
   $r_{child} \leftarrow r_{child} \ltimes r_v$



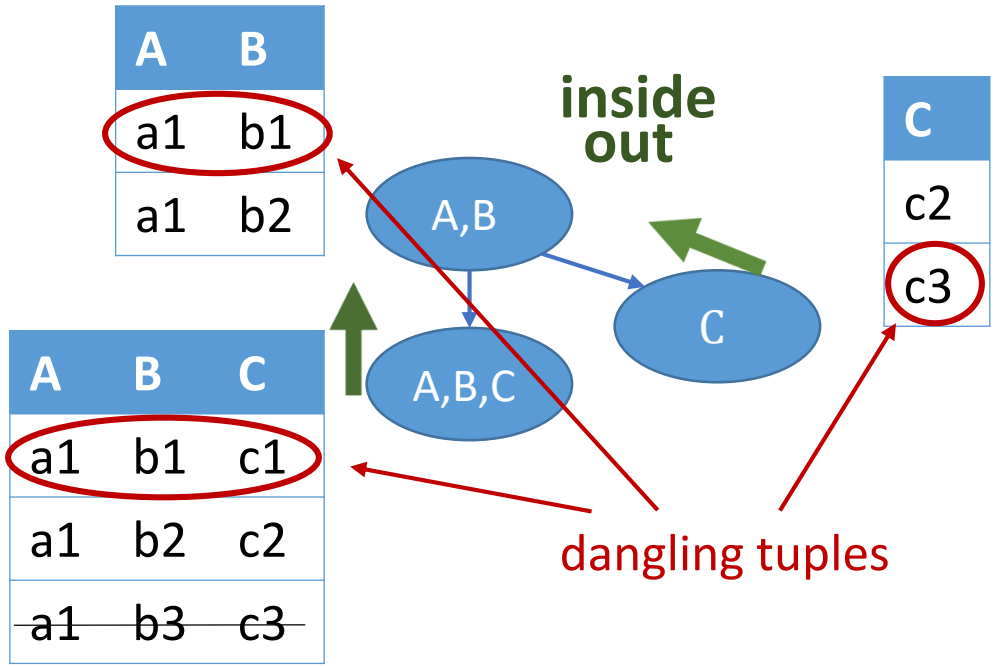
No dangling tuples!

Acyclic Joins

- An efficient algorithm for acyclic joins [Yannakakis81]
  1. Find a join tree and set a root
  2. Remove dangling tuples
  3. Join

only works with join trees

1. Leaf-to-root:
   $r_{parent} \leftarrow r_{parent} \ltimes r_v$
2. Root-to-leaf:
   $r_{child} \leftarrow r_{child} \ltimes r_v$



Acyclic Joins

- An efficient algorithm for acyclic joins [Yannakakis81]
  1. Find a join tree and set a root
  2. Remove dangling tuples
  3. Joi

only works with join trees

This approach fails for cyclic joins.
Is it possible with a different approach?

1. Lea
   r_{pa}
2. Roo
   $r_{child} \ltimes r_{child} \ltimes r_v$

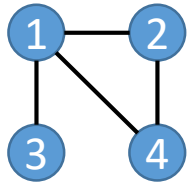
a1	b2	c2
~~a1~~	~~b3~~	~~c3~~

dangling tuples

Cyclic Joins

[Brault-Baron 2013]

Assumption: cannot detect triangles in a graph in linear time



edges

Q

x	y	z
1	2	4

$R_1 = R_2 = R_3$

1	2
1	3
1	4
2	4

Cyclic: $R_1(x, y) \bowtie R_2(y, z) \bowtie R_3(x, z)$

first answer in linear time $\Longrightarrow$ triangle in linear time $\Longrightarrow$ not possible

Cyclic Joins

[Brault-Baron 2013]

Assumption: k-Hypercliques cannot be found in time $O(m)$

m = number of edges of size $k-1$

edges

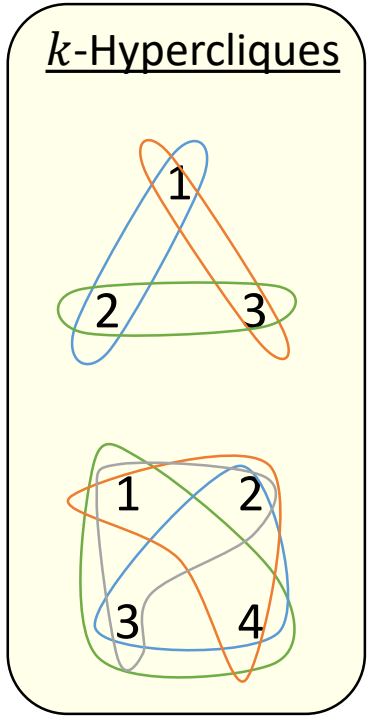
Q

x	y	z
1	2	4

$R_1 = R_2 = R_3$

1	2
1	3
1	4
2	4

Cyclic: $R_1(x,y) \bowtie R_2(y,z) \bowtie R_3(x,z)$



For every cyclic query, we can preform a similar reduction!

Joins Queries

- Given a join query Q,

If Q is acyclic, it is solvable
in linear preprocessing and constant delay

If Q is cyclic, a first answer
cannot be found in linear time *

* assuming hardness of k-hyperclique detection

Introduction Conclusions

- Some queries have easier structures than others.
- A dichotomy for natural join queries over general schemas.
- Other cases?

Research Question

What is the best we can do
with each query
given its structure?

Complexity of Queries

- Consider time complexity
- Treat every query as a problem
- Data complexity
  - DB = input
  - Query size = constant
- RAM model [Grandjean1996]
  - Lookup table: construction in linear time
    search in constant time
- Denote <lin,const>:
  queries solvable in linear time preprocessing and constant delay

Content

1. Intro: Join Queries and Acyclicity
2. Enumeration Complexity:
   - **Dependencies**
   - Unions
3. Additional Tasks:
   - Random Order
   - Ranked Access
   - Results: CQs, UCQs

Allowing projections (Conjunctive Queries)

$$Q(\text{Actor}, \text{Year}) \leftarrow cast(\text{Movie}, \text{Actor}) \wedge \text{release}(\text{Movie}, \text{Year})$$

Q:

Actor	Year
Richard Gere	1990
Julia Roberts	1990
Julia Roberts	2010
Tom Hanks	1994

Cast:

Movie	Actor
Pretty Woman	Richard Gere
Pretty Woman	Julia Roberts
Eat Pray Love	Julia Roberts
Forrest Gump	Tom Hanks

Release:

Movie	Year
Pretty Woman	1990
Eat Pray Love	2010
Forrest Gump	1994

Conjunctive Queries (CQs)

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

- Given a conjunctive query Q,

If Q is free-connex, $Q \in$ <lin,const>

If Q is acyclic not free-connex, $Q \notin$ <lin,const>*

If Q is cyclic, $Q \notin$ <lin,const>**

* no self-joins, assuming hardness of matrix multiplication

** no self-joins, assuming hardness of k-hyperclique detection

Definitions

An acyclic CQ has a graph with:

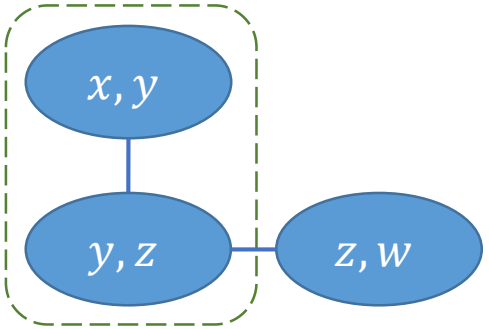
A free-connex CQ also requires:

1. a node for every atom
   possibly also subsets
2. tree
3. for every variable X:
   the nodes containing X form a subtree

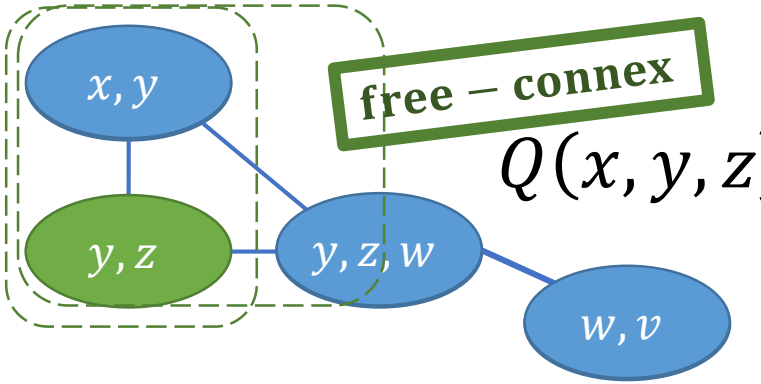
free − connex

acyclic

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



4. a subtree with exactly the free variables



free − connex

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

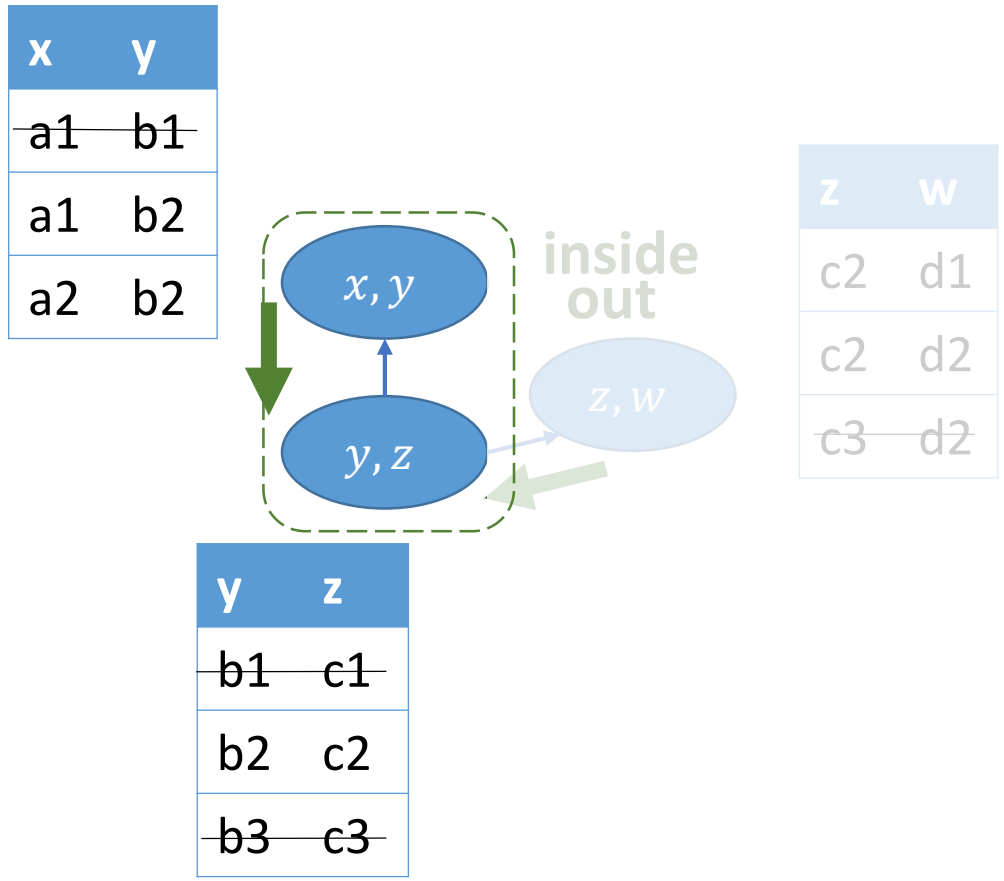
Free-Connex CQs

[BaganDurandGrandjean 2007]

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

Can be answered efficiently

1. Find a join tree
2. Remove dangling tuples [Yannakakis81]
3. Ignore existential variables
4. Join



CQs & DelayClin

Hard due to duplicates

Q	
1	1

R_1	
1	1
1	2
1	3

R_2	
1	1
2	1

Acyclic non-free-connex: $Q(x,z) \leftarrow R_1(x,y), R_2(y,z)$

Acyclic non-free-connex CQs [BaganDurandGrandjean CSL'2007]

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

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

Indices of ones

R_1

R	C
1	1
1	2
2	2

R_2

R	C
1	2
2	2

Q

R	C
1	2
2	2

Intractability cause:
$x - y - z$

Acyclic non-free-connex: $Q(x, z) \leftarrow R_1(x, y), R_2(y, z)$

$O(n^2)$ preprocessing + $O(1)$ delay = $O(n^2)$ total $\implies$ not possible

Conjunctive Queries

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

- Given a conjunctive query Q,

If Q is free-connex, $Q \in \mathrm{DelayC_{lin}}$

If Q is acyclic not free-connex, $Q \notin \mathrm{DelayC_{lin}}$ *

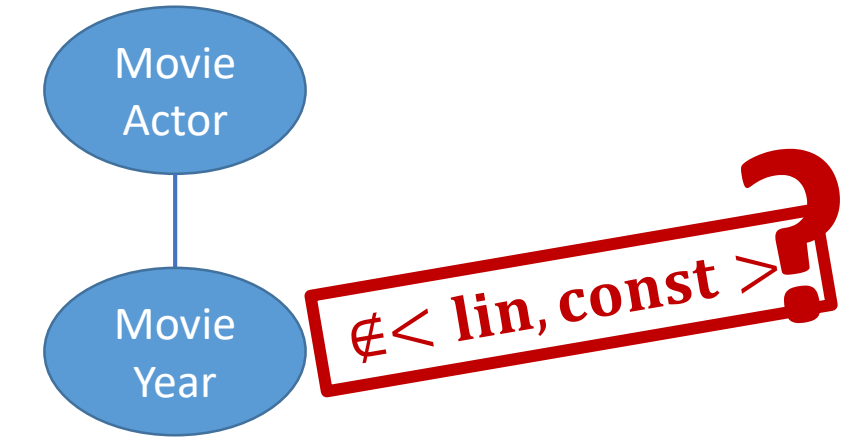
If Q is cyclic, $Q \notin \mathrm{DelayC_{lin}}$ *

* no self-joins, assuming hardness of matrix multiplication

** no self-joins, assuming hardness of k-hyperclique detection

Example

$$Q(\text{Actor}, \text{Year}) \leftarrow cast(\text{Movie}, \text{Actor}) \wedge \text{release}(\text{Movie}, \text{Year})$$



acyclic
not free-connex

Example

$$Q(\text{Actor}, \text{Year}) \leftarrow cast(\text{Movie}, \text{Actor}) \wedge release(\text{Movie}, \text{Year})$$

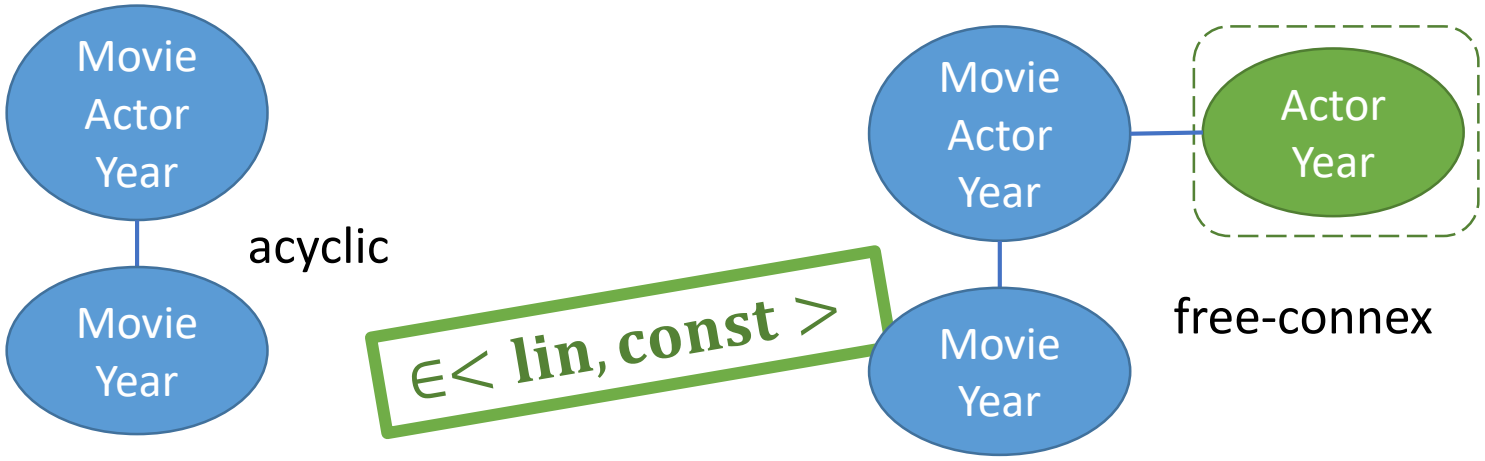
Cast:

Movie	Actor	Year
Pretty Woman	Richard Gere	1990
Pretty Woman	Julia Roberts	1990
Eat Pray Love	Julia Roberts	2010
Forrest Gump	Tom Hanks	1994

Release:

Movie	Year
Pretty Woman	1990
Eat Pray Love	2010
Forrest Gump	1994

$$Q^{+}(\text{Actor}, \text{Year}) \leftarrow cast(\text{Movie}, \text{Actor}, \text{Year}) \wedge release(\text{Movie}, \text{Year})$$



Example

$$Q(\text{Actor}, \text{Year}) \leftarrow cast(\text{Movie}, \text{Actor}) \wedge release(\text{Movie}, \text{Year})$$

Cast:

Movie	Actor	Year
Pretty Woman	Richard Gere	1990
Pretty Woman	Julia Roberts	1990
Eat Pray Love	Julia Roberts	2010
Forrest Gump	Tom Hanks	1994

Release:

Movie	Year
Pretty Woman	1990
Eat Pray Love	2010
Forrest Gump	1994

$$Q^{+}(\text{Actor}, \text{Year}) \leftarrow cast(\text{Movie}, \text{Actor}, \text{Year}) \wedge release(\text{Movie}, \text{Year})$$

- *Functional dependency* (*FD*):
  - A movie cannot have more than one release year
  - release: $1 \rightarrow 2$

$\in <$ lin, const $>$

Dependencies affect the complexity!

Acyclic non-free-connex CQs [BaganDurandGrandjean CSL'2007]

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

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

Indices of ones

R_1

R	C
1	1
1	2
2	2

R_2

R	C
1	2
2	2

Q

R	C
1	2
2	2

Intractability cause: $x - y - z$

If $x \to y$, we cannot perform this reduction

Acyclic non-free-connex: $Q(x, z) \leftarrow R_1(x, y), R_2(y, z)$

The hardness results do not hold with FDs

Conjunctive Queries

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

- Given a conjunctive query Q,

If Q is free-connex, $Q \in < lin, const >$

If Q is acyclic not free-connex, $Q \notin < lin, const >$*

If Q is cyclic, $Q \notin < lin, const >$*

* no self-joins, assuming hardness of matrix multiplication

** no self-joins, assuming hardness of k-hyperclique detection

Conjunctive Queries [C,Kröll; ICDT 2018, TOCS 2019]

- Given a conjunctive query Q **over a schema with unary FDs**,

If Q^+ is free-connex, $Q \in < lin, const >$

If Q^+ is acyclic not free-connex, $Q \notin < lin, const >$*

If Q^+ is cyclic, $Q \notin < lin, const >$*

Applies also for general FDs

* no self-joins, assuming hardness of matrix multiplication

** no self-joins, assuming hardness of k-hyperclique detection

Content

1. Intro: Join Queries and Acyclicity
2. Enumeration Complexity:
   - Dependencies
   - **Unions**
3. Additional Tasks:
   - Random Order
   - Ranked Access
   - Results: CQs, UCQs

Allowing Unions (UCQs)

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
Person	**Day**
Pablo Barceló	Mon
Martin Grohe	Mon
Muhammad Tibi	Mon

$$Q_1(\text{Person}, \text{Day}) \leftarrow \text{tutorials}(\text{Person}, \text{Title}), \text{schedule}(\text{Title}, \text{Day})$$
$$Q_2(\text{Person}, \text{Day}) \leftarrow \text{research}(\text{Person}, \text{Title}), \text{schedule}(\text{Title}, \text{Day})$$
$$Q_3 = Q_1 \cup Q_2$$

Cases for UCQs

All CQs are Easy

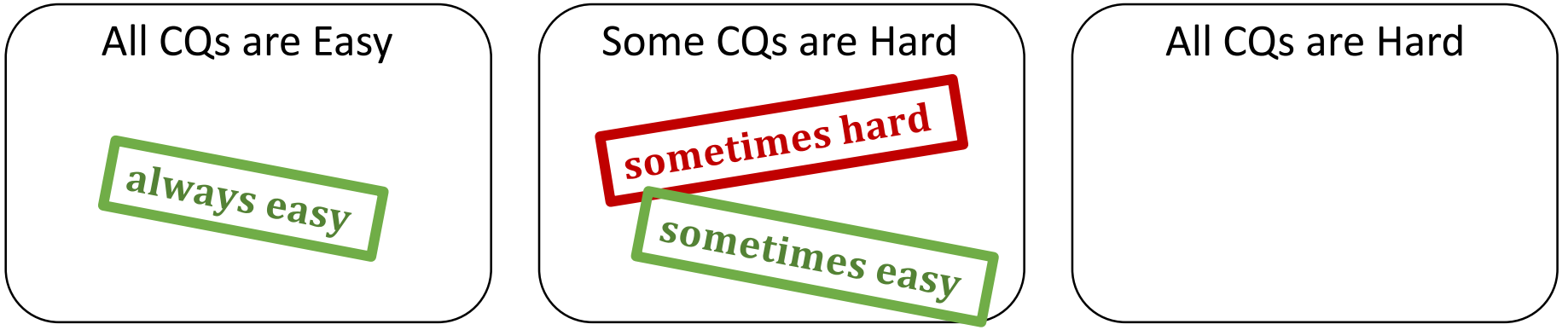
Some CQs are Hard

All CQs are Hard

If every CQ is easy, their union is easy too

Cases for UCQs

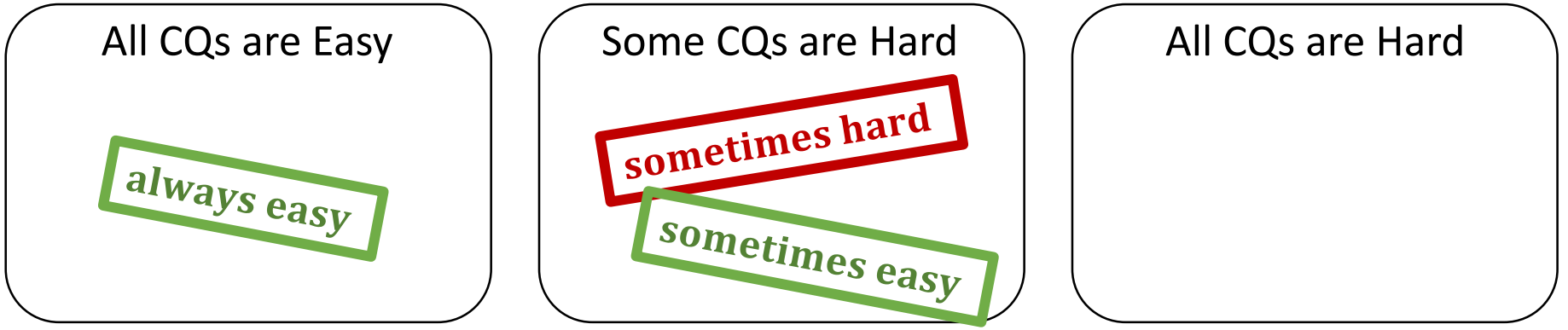
[C,Kröll; PODS 2019, TODS 2021]



Some non-redundant unions with a hard CQ are easy

Cases for UCQs

[C,Kröll; PODS 2019, TODS 2021]



If each CQ in Q is hard
and
there is no body-isomorphism

$\Longrightarrow \quad Q \notin DelayC_{lin}$

Cases for UCQs

[C,Kröll; PODS 2019, TODS 2021]

Some UCQs containing only hard CQs are easy!

Providing Variables

non free − connex

hard part

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

Body-homomorphism ↑ ↑ ∪ ↑ ↑

$$Q_2(a,b,c) \leftarrow R_1(a,b), R_2(b,c)$$

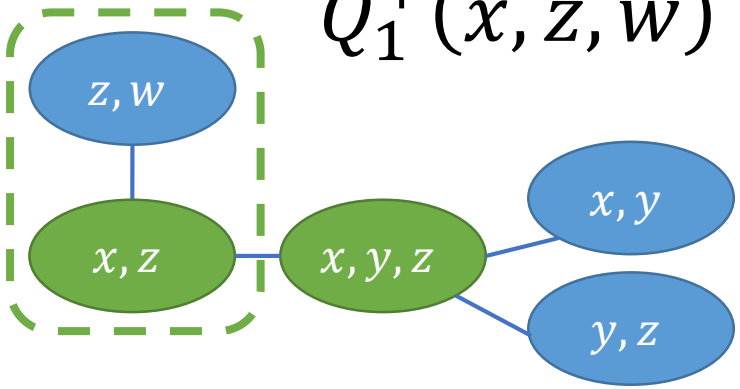
free − connex

$\in <$ lin, const $>$

Q_2 provides $\{x,y,z\}$ to Q_1

$$Q_1^+(x,z,w) \leftarrow R_1(x,y), R_2(y,z), R_3(z,w), R'(x,y,z)$$

free − connex



Cheater's Lemma

If an enumeration problem can be solved with:

- Usually constant delay
- Almost no duplicates

Then, it is $\in < \text{lin}, \text{const} >$

constant number of linear delay steps

constant number of duplicates per answer

Hard ∪ Hard = Easy

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

hard part

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

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

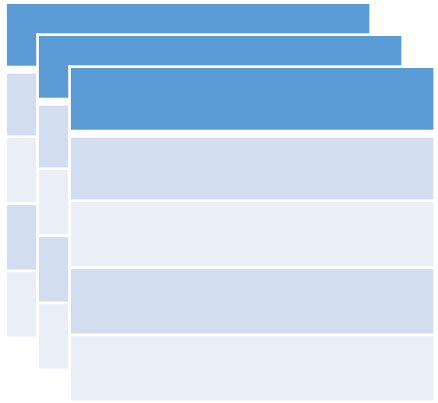
hard part

	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	

Content

1. Intro: Join Queries and Acyclicity
2. Enumeration Complexity:
   - Dependencies
   - Unions
3. Additional Tasks:
   - **Random Order**
   - Ranked Access
   - Results: CQs, UCQs

Why Random Permutation?



Database

+

Query

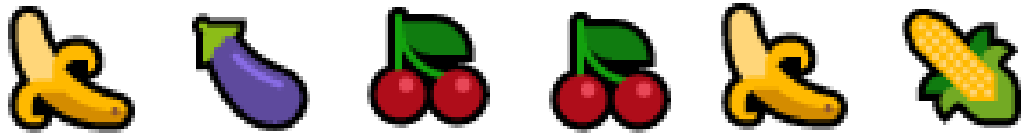
---

very large

Enumeration:

Downside: partial results not representative

Sampling:



Downside: repeating answers

Random Permutation:

Each answer once, uniformly random order

Idea: Separate the Task

- Find the number N of answers

6

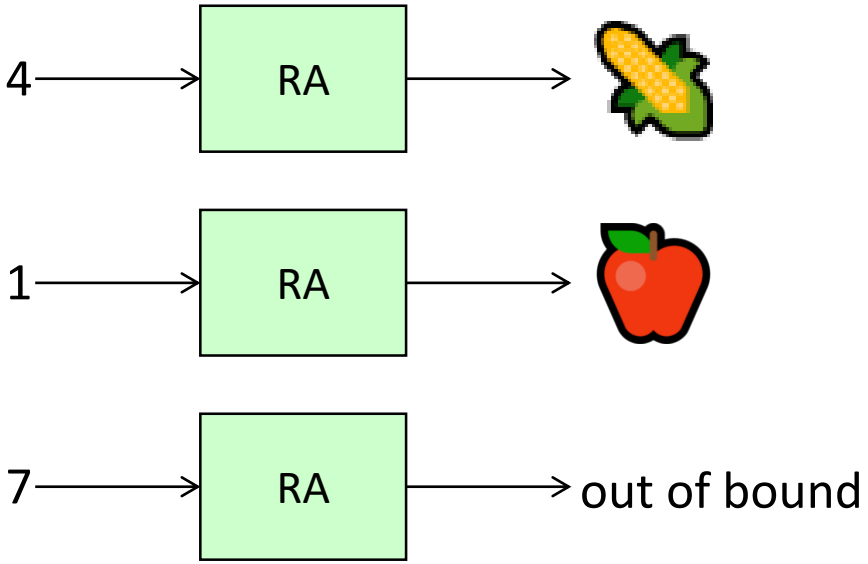
- Find a random permutation of 1,…,N

1 5 3 2 6 4

- Direct access to answers

Direct Access

- Simulates precomputed answers stored in an array
- Given i, returns the i^{th} answer or “out of bound”
- No constraints on the ordering used



Idea: Separate the Task

- Find the number N of answers

6

Via Direct Access

- Find a random permutation of 1,…,N

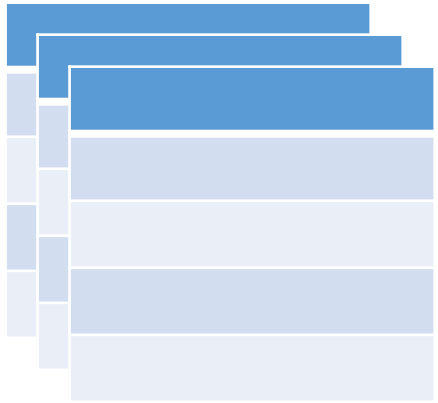
1 5 3 2 6 4

Modified Fisher-Yates Shuffle [Durstenfeld 1964]

- Direct access to answers

Direct Access

Consider 3 Tasks



Database

+

Query

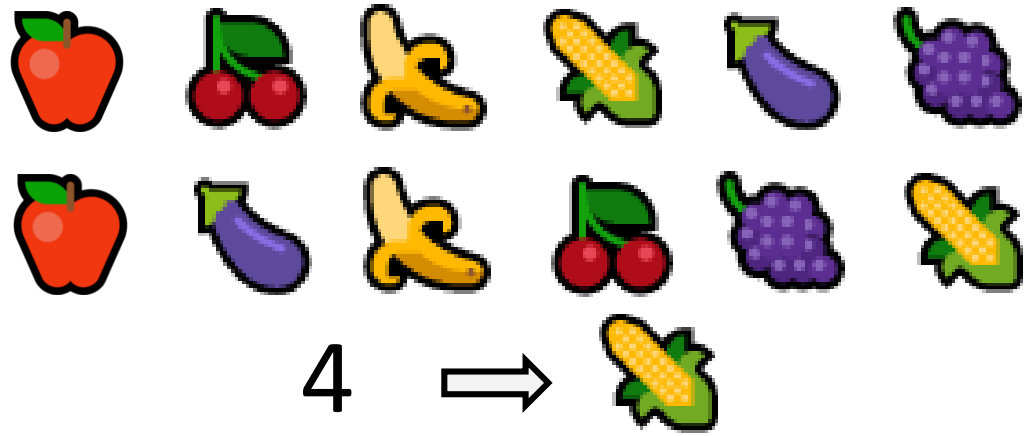
Enumeration:

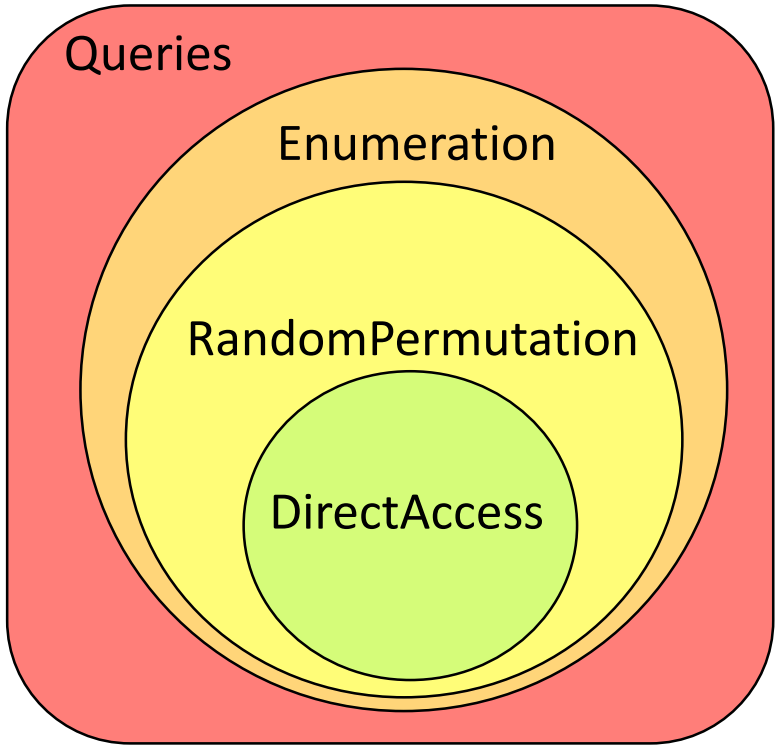
⇑

Random Permutation:

⇑

Direct Access:





Queries that can be computed with <lin,polylog>

Content

1. Intro: Join Queries and Acyclicity
2. Enumeration Complexity:
   - Dependencies
   - Unions
3. Additional Tasks:
   - Random Order
   - **Ranked Access**
   - Results: CQs, UCQs

Why Ranked Access?

Employees

Name	Role	Address
Jack	Junior dev	Boston
Jill	Senior dev	Brookline
Joanna	Senior dev	Braintree

Remuneration

Period	Role	Salary
11/2020	Junior dev	4000
11/2020	Senior dev	4500
12/2020	Junior dev	7000
12/2020	Senior dev	7100

Travel

Address	Cost
Boston	50
Brookline	100
Braintree	200

$Q(\text{Name}, \text{Role}, \text{Address}, \text{Period}, \text{Salary}, \text{Cost})$
$\leftarrow \text{Employees}(\text{Name}, \text{Role}, \text{Address}), \text{Remuneration}(\text{Period}, \text{Role}, \text{Salary}), \text{Travel}(\text{Role}, \text{Salary})$

Query Answers

sort by salary+cost

Name	Role	Address	Period	Salary	Cost
Jack	Junior dev	Boston	11/2020	4000	50
Jill	Senior dev	Brookline	11/2020	4500	100
Joanna	Senior dev	Braintree	11/2020	4500	200
Jack	Junior dev	Boston	12/2020	7000	50
…					

4^{th} result

Want:
- Median
- Boxplot
- Jump to any rank

without materializing all answers

Why Ranked Access?

Employees

Name	Role	Address
Jack	Junior dev	Boston
Jill	Senior dev	Brookline
Joan		

Remuneration

Period	Role	Salary
11/2020	Junior dev	4000
11/2020	Senior dev	4500

Travel

Address	Cost
Boston	50
Brookline	100

$Q(\text{Name, Ro}$
$\leftarrow \text{E}$ vel(Role, Salary)

Want:
Direct access with order

Qu

Na					
Jack	Junior dev	Boston	11/2020	4000	50
Jill	Senior dev	Brookline	11/2020	4500	100
Joanna	Senior dev	Braintree	11/2020	4500	200
Jack	Junior dev	Boston	12/2020	7000	50
…					

sort by salary+cost

- Boxplot
- Jump to any rank

without materializing all answers

4th result

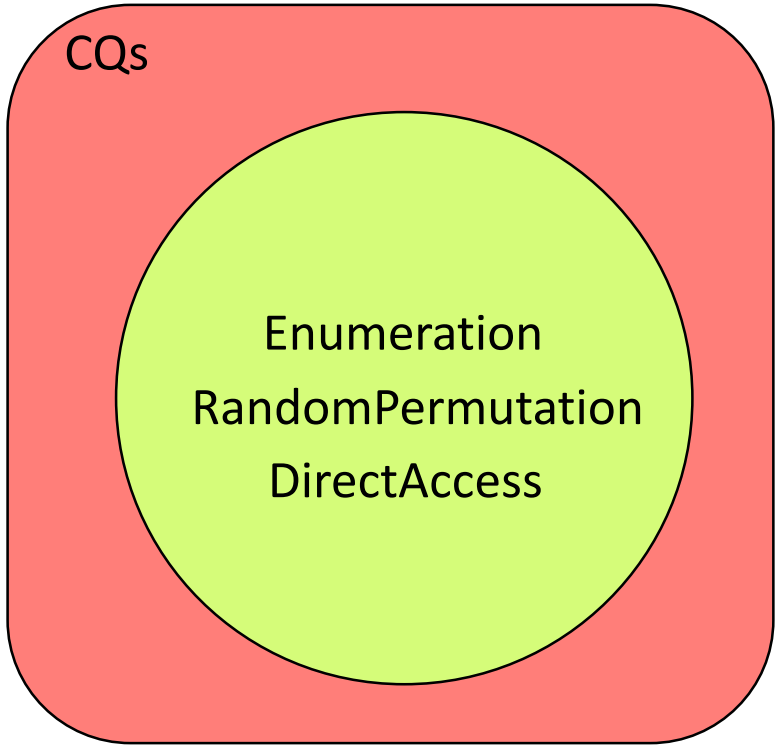
Content

1. Intro: Join Queries and Acyclicity
2. Enumeration Complexity:
   - Dependencies
   - Unions
3. Additional Tasks:
   - Random Order
   - Ranked Access
   - **Results: CQs, UCQs**

Results for CQs

[**C**,Zeevi,Berkholz,Kimelfeld,Schweikardt; PODS 2020]
[**C**,Tziavelis,Gatterbauer,Kimelfeld,Riedewald; PODS 2021]

- Consider <lin,log>:
- Direct access:
  - Tractable iff* free-connex

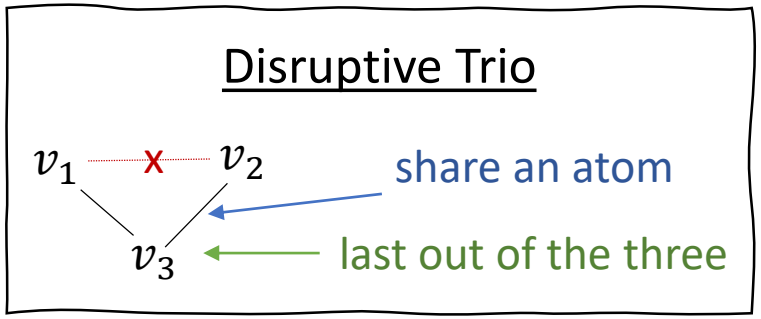


* Lower bounds under the assumptions from before

Results for CQs

[C,Zeevi,Berkholz,Kimelfeld,Schweikardt; PODS 2020]
[C,Tziavelis,Gatterbauer,Kimelfeld,Riedewald; PODS 2021]

- Consider <lin,log>:
- Direct access:
  - Tractable iff* free-connex
- Direct Access – ranked by lexicographic order:
  - Tractable iff* free-connex and no disruptive trio
- Direct Access – ranked by sum of weights order:
  - Tractable iff* free-connex and free variables in one atom



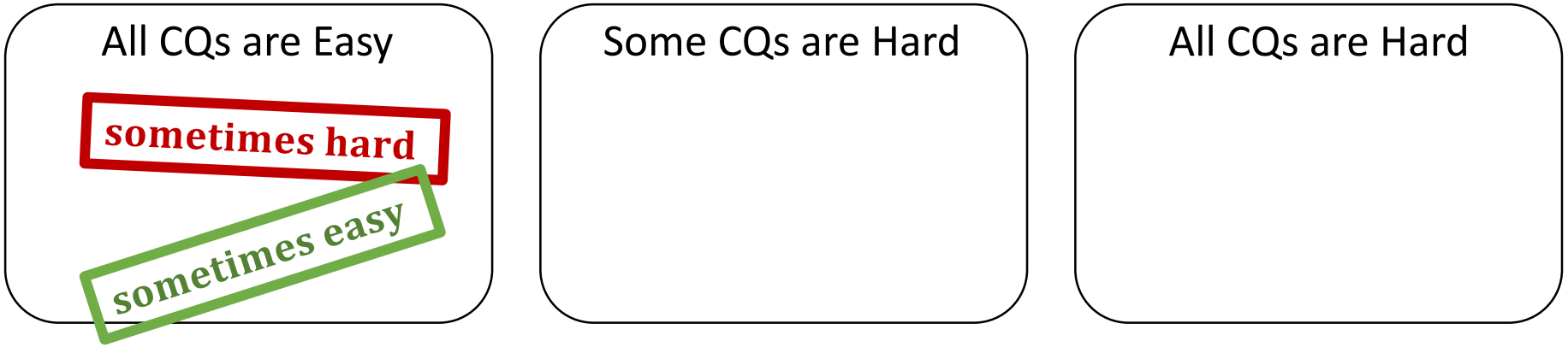
$$Q(v_1, v_2, v_3) \leftarrow R(v_1, v_3), S(v_3, v_2)$$

* Lower bounds under the assumptions from before and the 3-SUM hypothesis

Results for UCQs

- Enumeration:

- Direct Access:



Results for UCQs

Example: Easy ∪ Easy = Sometimes Hard

free − connex $Q_1(x, y, z) \leftarrow R(x, y), S(y, z)$

free − connex $Q_2(x, y, z) \leftarrow S(y, z), T(x, z)$

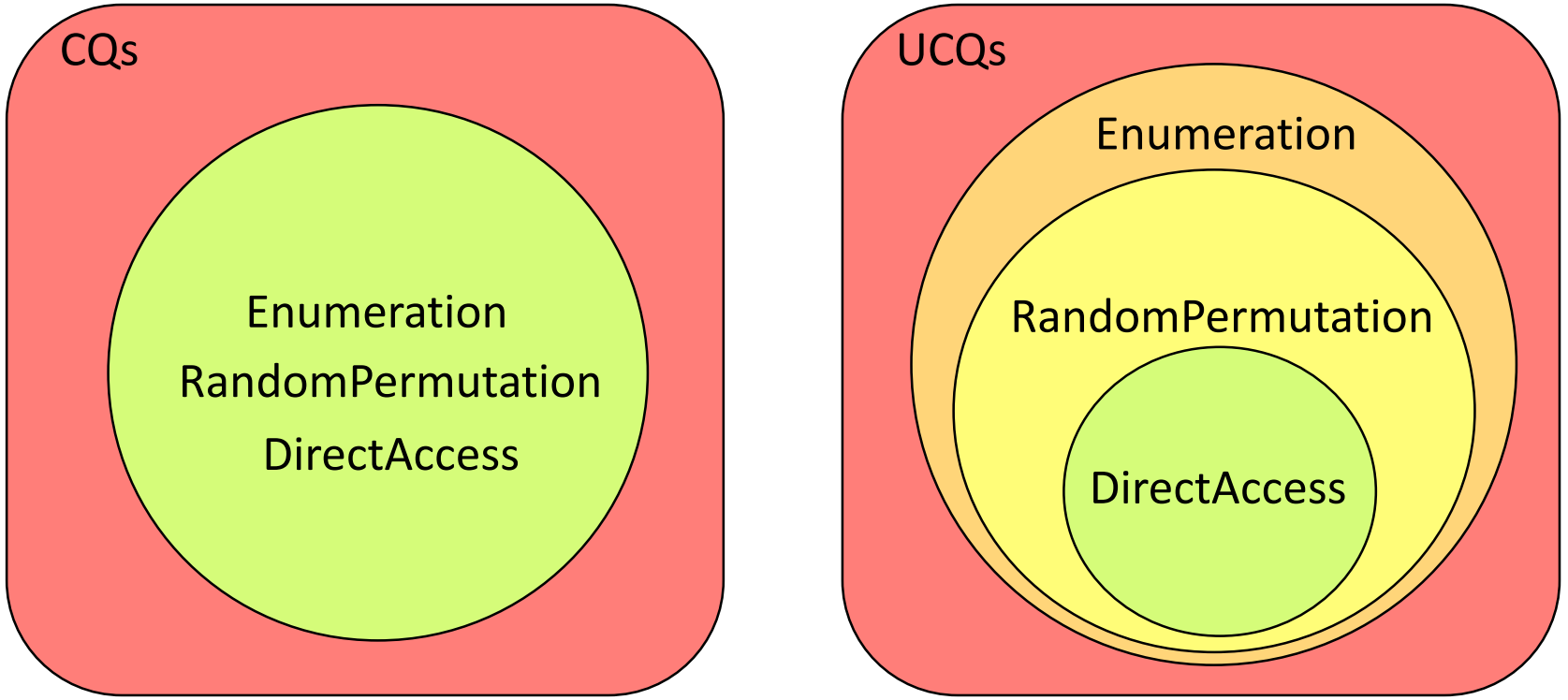
cyclic $Q_1 \cap Q_2(x, y, z) \leftarrow R(x, y), S(y, z), T(x, z)$

- Cannot count $Q_1 \cap Q_2$ in linear time
  * assumption: cannot detect a triangle in a graph in linear time.
- If $Q_1 \cup Q_2$ has efficient direct access,
  - Can count $|Q_1 \cup Q_2|$ in linear time
  - Can compute $|Q_1 \cap Q_2| = |Q_1| + |Q_2| - |Q_1 \cup Q_2|$

Contradiction!

Comparing the Tasks

- UCQs: Enumeration ⇏ RandomAccess



- Alternatives for random permutation for UCQs

[**C**,Zeevi,Berkholz,Kimelfeld,Schweikardt; PODS 2020]

Content

1. Intro: Join Queries and Acyclicity
2. Enumeration Complexity:
   - Dependencies
   - Unions
3. Additional Tasks:
   - Random Order
   - Ranked Access
   - Results: CQs, UCQs

What’s next?

- Dichotomies
- Queries with self-joins
- Larger query classes
- Direct access without counting
- Tools for enumeration lower bounds
- Tradeoff preprocessing-delay for hard cases
- Space consumption
- Applicability in practice

More info: carme.li