Enumeration Complexity of CQs with Functional Dependencies

Nofar Carmeli Markus Kröll

Content

- Settings & Motivation
- Our Contribution
- Future Work

Conjunctive Queries (CQs) - Example

Cast:

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

Release:

Movie	Budget
Pretty Woman	14 million
Eat Pray Love	60 million
Forrest Gump	55 million

Q:

Actor	Budget
Richard Gere	14 million
Julia Roberts	14 million
Julia Roberts	60 million
Tom Hanks	55 million

Q: a list of actors and the budgets of movies in which they participated

$$\mathrm{Q}(\mathrm{Actor}, \mathrm{Budget}) \leftarrow cast(\mathrm{Movie}, \mathrm{Actor}) \wedge \mathrm{release}(\mathrm{Movie}, \mathrm{Budget})$$

$$Q(\blacksquare\bullet) \leftarrow R_1(\blacktriangle\blacksquare), R_2(\blacktriangle\bullet)$$

Complexity of CQs

- Data complexity
  - Input: DB instance
  - The query is considered constant

- RAM model [Grandjean1996]
  - Lookup table: construction in linear time
    search in constant time

What is the best we can hope for?

Complexity of CQs

- Examples:
  1. Cartesian product

1
2
3
4

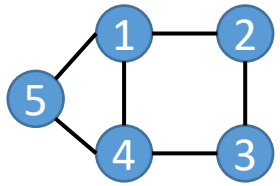
a
b
c

u
v

$$\mathrm{Q}(x, y, z) \leftarrow R(x) \wedge S(y) \wedge T(z)$$

1	a	u
1	a	v
1	b	u
1	b	v
1	c	u
…	…	…

  2. Finding triangles



1	2
1	4
1	5
…	…

$$\mathrm{Q}(x, y, z) \leftarrow R(x, y) \wedge R(y, z) \wedge R(x, z)$$

- Ideally:
  - linear scan before first (to read input)
  - constant delay between answers (to write results)

Complexity of CQs

- Ideally:
  - linear scan before first (to read input)
  - constant delay between answers (to write results)
- Can build a compact representation during the initial scan.

Can this always be done? No.

When can it be done?

$DelayC_{lin}$: solvable with linear time preprocessing and constant delay

Which CQs are in $DelayC_{lin}$?

Known Dichotomy [BaganDurandGrandjean CSL'2007]

Dichotomy

Self-join-free acyclic queries are

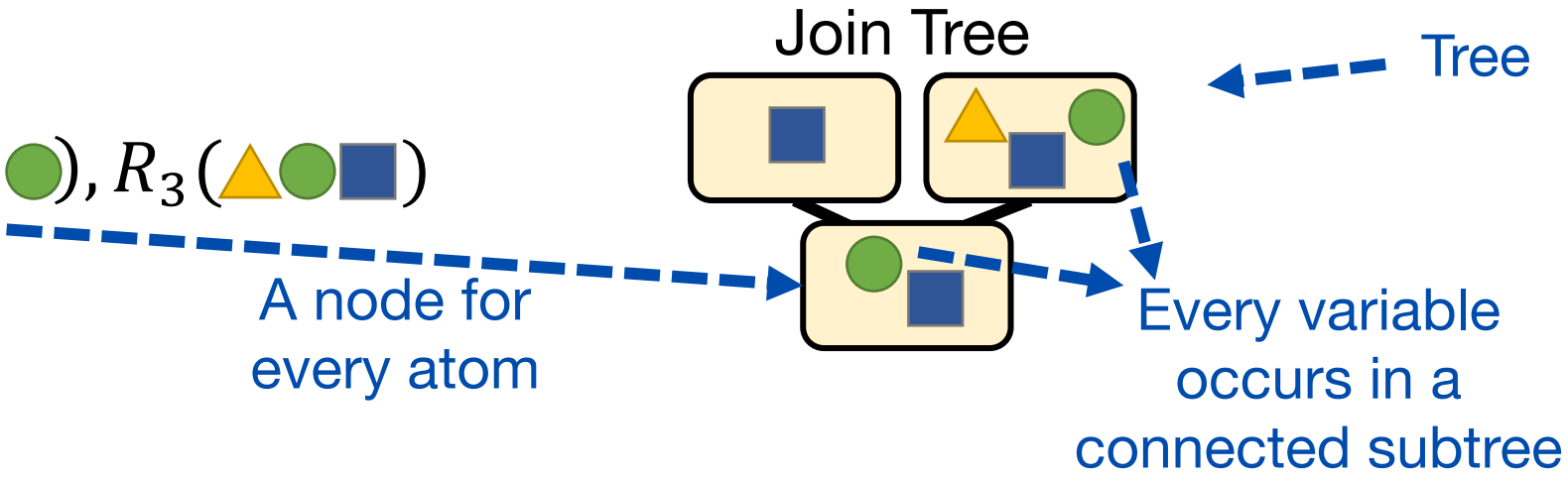
$$\in DelayC_{lin} \Leftrightarrow \text{free-connex}$$

* assumption: Boolean matrix multiplication cannot be done in quadratic time

Join Tree

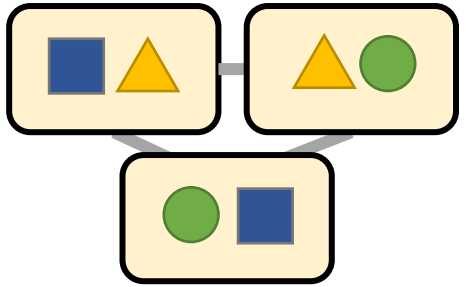
Query

$Q(\,) \leftarrow R_1(\blacksquare), R_2(\blacksquare\bullet), R_3(\blacktriangle\bullet\blacksquare)$



- A query may have no join tree

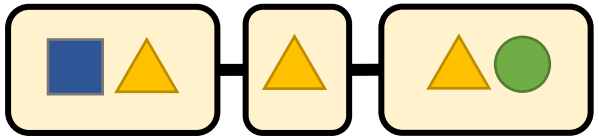
$Q(\,) \leftarrow R_1(\blacksquare\blacktriangle), R_2(\blacktriangle\bullet), R_3(\bullet\blacksquare)$



- A query that has a join tree is called acyclic

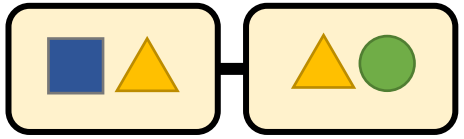
CQ Classes

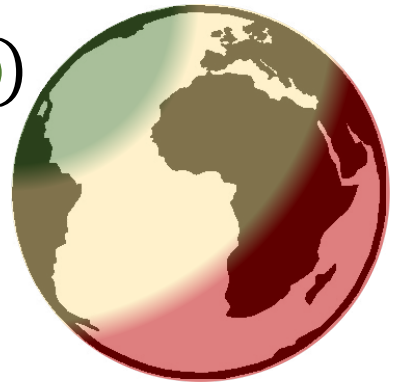
- Acyclic: has a join tree
- Free-connex: has a join tree including the head



acyclic free-connex:

$Q(\blacktriangle) \leftarrow R_1(\blacksquare\blacktriangle), R_2(\blacktriangle\bullet)$



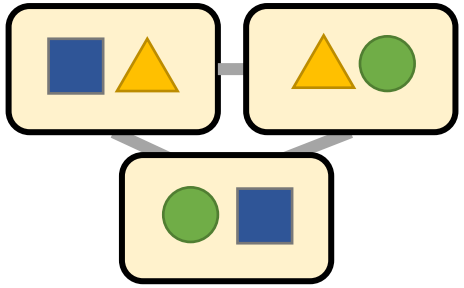


cyclic:

$Q(\,) \leftarrow R_1(\blacksquare\blacktriangle), R_2(\blacktriangle\bullet), R_3(\bullet\blacksquare)$

acyclic non-free-connex:

$Q(\bullet\blacksquare) \leftarrow R_1(\blacksquare\blacktriangle), R_2(\blacktriangle\bullet)$



Known Dichotomy [BaganDurandGrandjean CSL'2007]

Self-join-free acyclic queries are

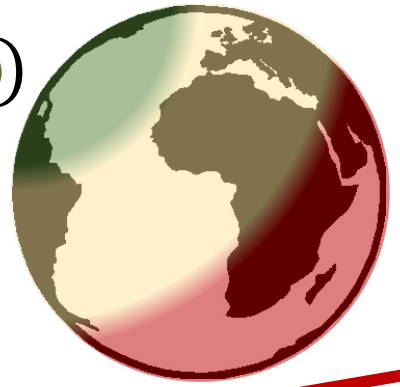
$$\in DelayC_{lin} \Leftrightarrow \text{free-connex}$$

* assumption: Boolean matrix multiplication cannot be done in quadratic time

acyclic free-connex:

$Q(\blacktriangle) \leftarrow R_1(\blacksquare\blacktriangle), R_2(\blacktriangle\bullet)$

$\in$ DelayC$_{lin}$



cyclic:

$Q(\,) \leftarrow R_1(\blacksquare\blacktriangle), R_2(\blacktriangle\bullet), R_3(\bullet\blacksquare)$

acyclic non-free-connex:

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

$Q(\bullet\blacksquare) \leftarrow R_1(\blacksquare\blacktriangle), R_2(\blacktriangle\bullet)$

* no self joins, under the matrix multiplication assumption

Known Dichotomy [Brault-Baron 2013]

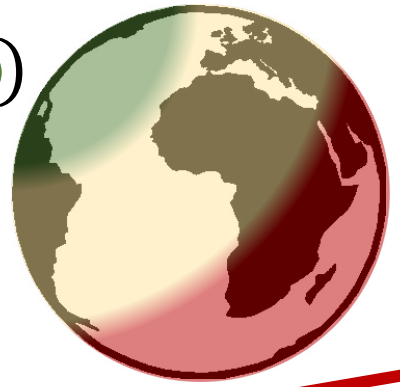
Self-join-free cyclic queries cannot be decided in linear time

* assumption: the existence of a Tetra structure in a graph cannot be decided in linear time

acyclic free-connex:

$Q(\blacktriangle) \leftarrow R_1(\blacksquare\blacktriangle), R_2(\blacktriangle\bullet)$

$\in \mathbf{DelayC_{lin}}$



cyclic:

$\notin \mathbf{DelayC_{lin}}^{**}$

$Q() \leftarrow R_1(\blacksquare\blacktriangle), R_2(\blacktriangle\bullet), R_3(\bullet\blacksquare)$

acyclic non-free-connex:

$\notin \mathbf{DelayC_{lin}}^{*}$

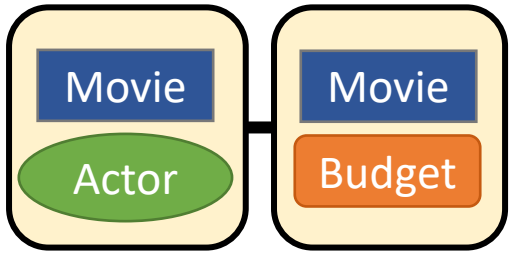
$Q(\bullet\blacksquare) \leftarrow R_1(\blacksquare\blacktriangle), R_2(\blacktriangle\bullet)$

* no self joins, under the matrix multiplication assumption

** no self joins, under the Tetra assumption

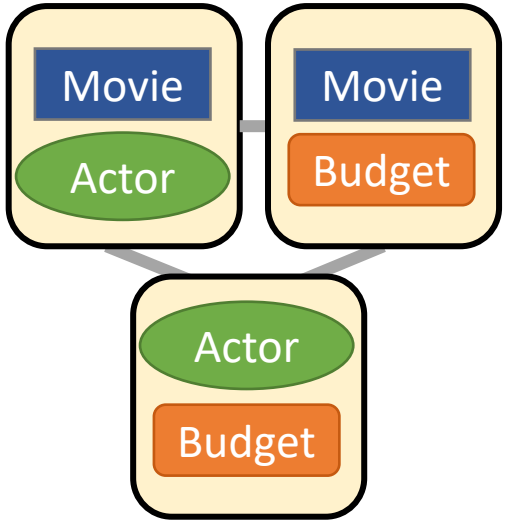
Example

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



acyclic

$\notin \mathbf{DelayC_{lin}}$?



not free-connex

Example

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

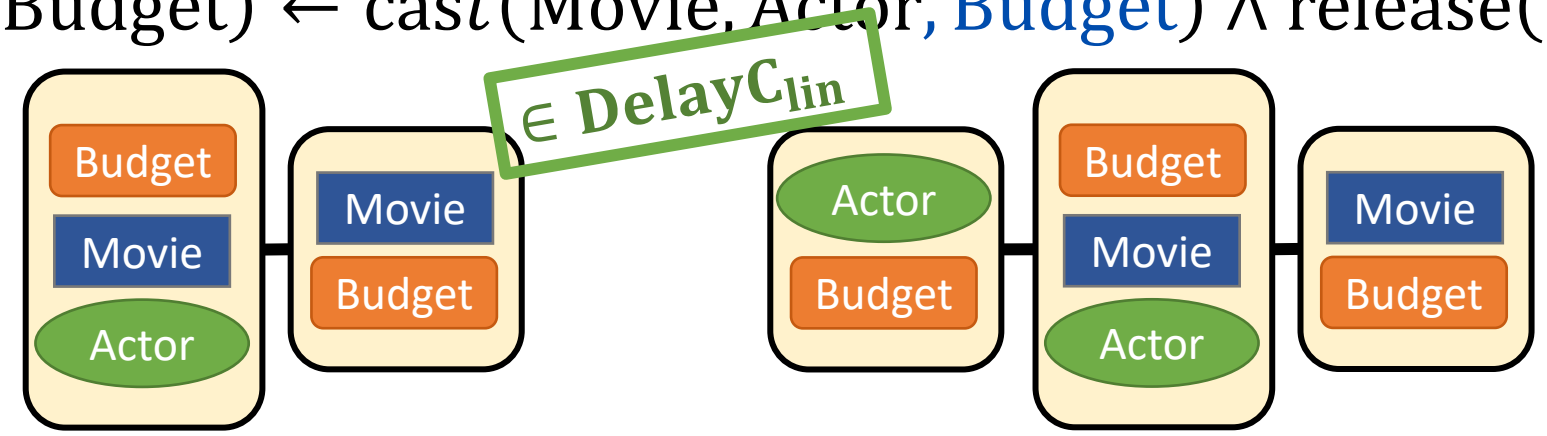
Cast:

Movie	Actor	Budget
Pretty Woman	Richard Gere	14 million
Pretty Woman	Julia Roberts	14 million
Eat Pray Love	Julia Roberts	60 million
Forrest Gump	Tom Hanks	55 million

Release:

Movie	Budget
Pretty Woman	14 million
Eat Pray Love	60 million
Forrest Gump	55 million

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



Example

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

Cast:

Movie	Actor	Budget
Pretty Woman	Richard Gere	14 million
Pretty Woman	Julia Roberts	14 million
Eat Pray Love	Julia Roberts	60 million
Forrest Gump	Tom Hanks	55 million

Release:

Movie	Budget
Pretty Woman	14 million
Eat Pray Love	60 million
Forrest Gump	55 million

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

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

Example

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

release: $1 \rightarrow 2$

$\in \mathbf{DelayC_{lin}}$

But the dichotomy said $Q \notin \mathbf{DelayC_{lin}}$!

- Not really.
- The hardness result is based on a reduction $\text{Enum}\langle\Pi\rangle \leq_e \text{Enum}_{\Delta}\langle Q\rangle$
- The reduction may assign release any combination of tuples
- It doesn't apply with FDs

Our Goal

Classification of CQ/FDs combinations
w.r.t. $DelayC_{lin}$

Content

- Settings & Motivation
- Our Contribution
- Future Work

Method

- Define the extended query
- Classify according to the extension

Treat the FDs as between variables
Apply the FDs to all relevant atoms(+head)

$$Q(x) \leftarrow R_1(x,y), R_2(x,z), R_3(w,y,z)$$
$$R_1: 1 \rightarrow 2$$
$$R_3: 2,3 \rightarrow 1$$

$x \rightarrow y$ ↓ Q

$$Q'(x,y) \leftarrow R_1(x,y), R_2(x,z), R_3(w,y,z)$$
$$R_1: 1 \rightarrow 2$$
$$R_3: 2,3 \rightarrow 1$$

R_2 → $x \rightarrow y$

$$Q^+(x,y) \leftarrow R_1(x,y), R_2(x,z,y,w), R_3(w,y,z)$$
$$R_1: 1 \rightarrow 2$$
$$R_3: 2,3 \rightarrow 1$$
$$R_2: 1 \rightarrow 3$$
$$R_2: 3,2 \rightarrow 4$$

R_2 ↑ $yz \rightarrow w$

$$Q''(x,y) \leftarrow R_1(x,y), R_2(x,z,y), R_3(w,y,z)$$
$$R_1: 1 \rightarrow 2$$
$$R_3: 2,3 \rightarrow 1$$
$$R_2: 1 \rightarrow 3$$

$$x \rightarrow y$$
$$yz \rightarrow w$$

Method

- Define the extended query 
- Classify according to the extension 

Treat the FDs as between variables
Apply the FDs to all relevant atoms(+head)

$$Q(x) \leftarrow R_1(x,y), R_2(x,z), R_3(w,y,z)$$
$$R_1: 1 \rightarrow 2$$
$$R_3: 2,3 \rightarrow 1$$

$$Q^+(x,y) \leftarrow R_1(x,y), R_2(x,z,y,w), R_3(w,y,z)$$
$$R_1: 1 \rightarrow 2$$
$$R_3: 2,3 \rightarrow 1$$
$$R_2: 1 \rightarrow 3$$
$$R_2: 3,2 \rightarrow 4$$

Known Results

acyclic free-connex: $\in \mathbf{DelayC_{lin}}$

$Q(\blacktriangle) \leftarrow R_1(\blacksquare\blacktriangle), R_2(\blacktriangle\bullet)$

cyclic: $\notin \mathbf{DelayC_{lin}}^{**}$

$Q(\,) \leftarrow R_1(\blacksquare\blacktriangle), R_2(\blacktriangle\bullet), R_3(\bullet\blacksquare)$

acyclic non-free-connex: $\notin \mathbf{DelayC_{lin}}^{*}$

$Q(\bullet\blacksquare) \leftarrow R_1(\blacksquare\blacktriangle), R_2(\blacktriangle\bullet)$

* no self joins, under the matrix multiplication assumption

** no self joins, under the Tetra assumption

Our Results

acyclic free-connex: $Q^+(\blacktriangle) \leftarrow R_1(\blacksquare\blacktriangle), R_2(\blacktriangle\bullet)$

$Q \in \mathbf{DelayC_{lin}}$

cyclic: $Q^+(\,) \leftarrow R_1(\blacksquare\blacktriangle), R_2(\blacktriangle\bullet), R_3(\bullet\blacksquare)$

$Q \notin \mathbf{DelayC_{lin}}^{**}$

acyclic non-free-connex: $Q^+(\bullet\blacksquare) \leftarrow R_1(\blacksquare\blacktriangle), R_2(\blacktriangle\bullet)$

$Q \notin \mathbf{DelayC_{lin}}^{*}$

* no self joins, under the matrix multiplication assumption

** no self joins, **only unary FDs,** under the Tetra assumption

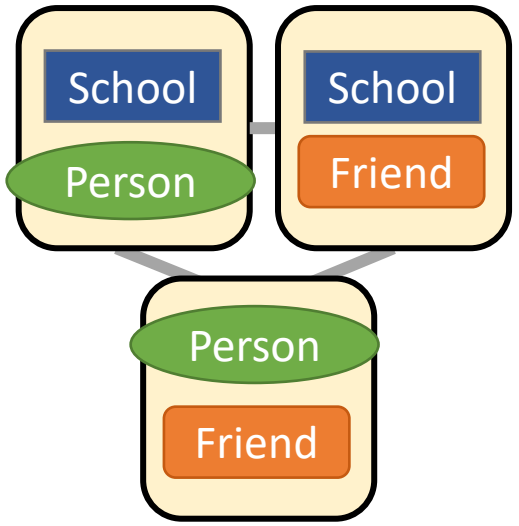
Another Example

$Q(\text{Person}, \text{Friend}) \leftarrow$
$\text{friends}(\text{Person}, \text{Friend}) \wedge \text{students}(\text{Person}, \text{School}) \wedge \text{students}(\text{Friend}, \text{School})$

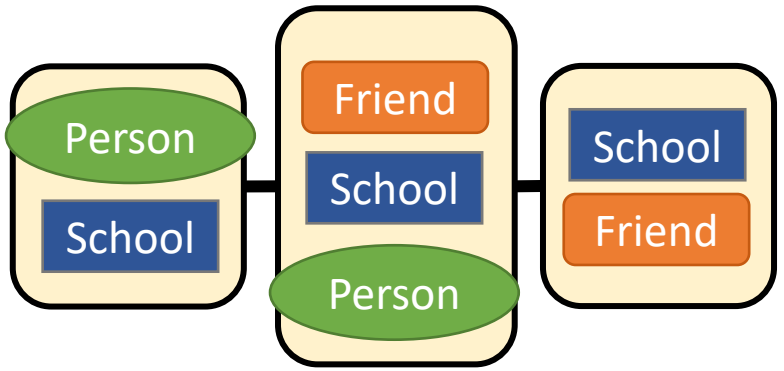
$\text{students}: \text{Person} \rightarrow \text{School}$

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

$Q(\text{Person}, \text{Friend}, \text{School}) \leftarrow \text{friends}(\text{Person}, \text{Friend}, \text{School}) \wedge$
$\text{students}(\text{Person}, \text{School}) \wedge \text{students}(\text{Friend}, \text{School})$



Q cyclic



Q^+ free-connex acyclic

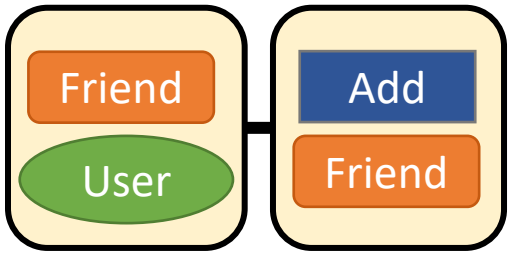
Another Example

$$Q(\text{User}, \text{Ad}) \leftarrow \text{friends}(\text{User}, \text{Friend}) \wedge \text{clicked}(\text{Friend}, \text{Ad})$$

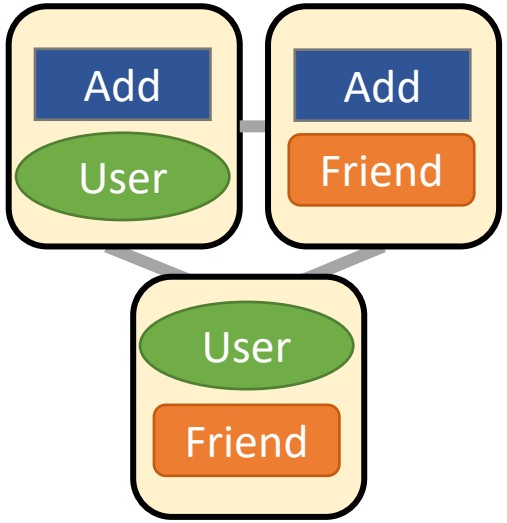
- *Cardinality dependency*:
  - A user cannot have more than 5000 friends
  - $\text{friends}(\text{User} \rightarrow \text{Friend}, 5000)$

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

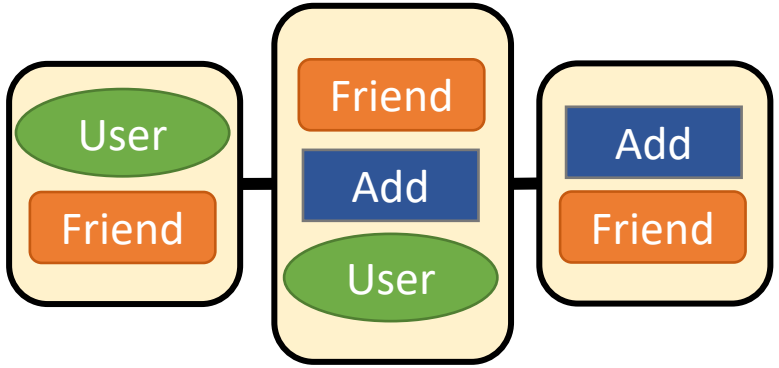
$$Q^{+}(\text{User}, \text{Ad}, \text{Friend}) \leftarrow \text{friends}(\text{User}, \text{Friend}) \wedge \text{clicked}(\text{Friend}, \text{Ad})$$



acyclic



Q not free-connex



Q^{+} free-connex

Our Results

acyclic free-connex:

$Q^+(\blacktriangle) \leftarrow R_1(\blacksquare\blacktriangle), R_2(\blacktriangle\bullet)$

$Q \in \mathbf{DelayC_{lin}}$

cyclic:

$Q^+(\,) \leftarrow R_1(\blacksquare\blacktriangle), R_2(\blacktriangle\bullet), R_3(\bullet\blacksquare)$

$Q \notin \mathbf{DelayC_{lin}}^{**}$

Reduce Q to Q^+

acyclic non-free-connex:

$Q^+(\bullet\blacksquare) \leftarrow R_1(\blacksquare\blacktriangle), R_2(\blacktriangle\bullet)$

$Q \notin \mathbf{DelayC_{lin}}$

* no self joins, under the matrix multiplication assumption

** no self joins, **only unary FDs,** under the Tetra assumption

Our Results

acyclic free-connex:

$Q^+(\blacktriangle) \leftarrow R_1(\blacksquare\blacktriangle), R_2(\blacktriangle\bullet)$

$\mathbf{Q \in DelayC_{lin}}$

Reduce Q^+ to Q

Show that a non-free-connex Q^+ is hard

$, R_2(\blacktriangle\bullet), R_3(\bullet\blacksquare)$

$\mathbf{Q \notin DelayC_{lin}}^*$

acyclic non-free-connex:

$Q^+(\bullet\blacksquare) \leftarrow R_1(\blacksquare\blacktriangle), R_2(\blacktriangle\bullet)$

* no self joins, under the matrix multiplication assumption

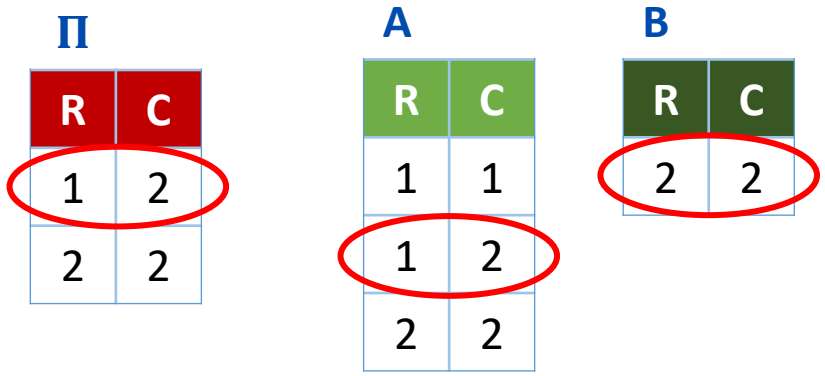
** no self joins, **only unary FDs,** under the Tetra assumption

Proof [BaganDurandGrandjean CSL'2007]

- Assumption: Boolean $n \times n$ matrix multiplication cannot be done in time $O(n^2)$

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

- Therefore, answering $\Pi(a,b) \leftarrow A(a,c) \wedge B(c,b)$ is not in $DelayC_{lin}$



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

Reduction

$\Pi(x, y) \leftarrow A(x, z) \wedge B(z, y)$ is not in $DelayC_{lin}$

Π

R	C
1	2
2	2

A

R	C
1	1
1	2
2	2

B

R	C
2	2

Problem in case $R_1: v \rightarrow x$

Solution: assign v with the same values as x

Reduction

$Q(x, y, u) \leftarrow R_1(x, v), R_2(x, z_1, u, t), R_3(z_1, z_2, u), R_4(z_2, y, u)$

Q

x	y	u
1	2	⊥
2	2	⊥

R_1

x	v
1	**1**
2	**2**

R_2

x	z1	u	t
1	1	⊥	⊥
1	2	⊥	⊥
2	2	⊥	⊥

R_3

z1	z2	u
1	1	⊥
2	2	⊥

R_4

z2	y	u
2	2	⊥

Reduction

$\Pi(x, y) \leftarrow A(x, z) \wedge B(z, y)$ is not in $DelayC_{lin}$

Π

R	C
1	2
2	2

A

R	C
1	1
1	2
2	2

B

R	C
2	2

Problem in case R_2: $ut \rightarrow x$

Solution: assign both u and t the same value as x ?

Reduction

$Q(x, y, u) \leftarrow R_1(x, v), R_2(x, z_1, u, t), R_3(z_1, z_2, u), R_4(z_2, y, u)$

Q

x	y	u
1	2	**1**
2	2	**2**

R_1

x	v
1	⊥
2	⊥

R_2

x	z1	u	t
1	1	**1**	**1**
1	2	**1**	**1**
2	2	**2**	**2**

R_3

z1	z2	u
1	1	⊥
2	2	**2**

R_4

z2	y	u
2	2	**2**

Reduction

$\Pi(x, y) \leftarrow A(x, z) \wedge B(z, y)$ is not in $DelayC_{lin}$

Π

R	C
1	2
2	2

A

R	C
1	1
1	2
2	2

B

R	C
2	2

Problem in case R_2: $ut \rightarrow x$

Solution: only assign t the same value as x

Reduction

$Q(x, y, u) \leftarrow R_1(x, v), R_2(x, z_1, u, t), R_3(z_1, z_2, u), R_4(z_2, y, u)$

Q

x	y	u
1	2	⊥
2	2	⊥

R_1

x	v
1	⊥
2	⊥

R_2

x	z1	u	t
1	1	⊥	**1**
1	2	⊥	**1**
2	2	⊥	**2**

R_3

z1	z2	u
1	1	⊥
2	2	⊥

R_4

z2	y	u
2	2	⊥

Reduction

Is it **always** possible to adjust this reduction s.t.

- Construction in linear time
- One-one mapping of answers
- FDs hold

?

If it is an FD-extension, we prove that it is

!

In General

- In the presence of FDs, **more queries** are tractable
- We show how to use FD-extensions to:
  - Show **tractability** of additional queries
  - Adjust **hardness** results to apply with FDs

Content

- Settings & Motivation
- Our Contribution
- Future Work

Future Work

- Completing the dichotomy
  - Show Q^+ is cyclic ⇒ Q not in $DelayC_{lin}$ for general FDs
- A dichotomy for negated queries
  - Negated acyclic queries can be answered with logarithmic delay after quasilinear time preprocessing iff they are free-connex signed-acyclic [Brault-Baron2013]
- Using a weaker complexity assumption for the cyclic case
- Richer query classes
  - Remove the no self joins assumption


THANK
YOU.
THANK
YOU VERY
MUCH.