Finding Consistent Answers from Inconsistent Data: Systems, Algorithms, and Complexity

Xiating Ouyang

University of Wisconsin–Madison

PhD Defense, November 21, 2023

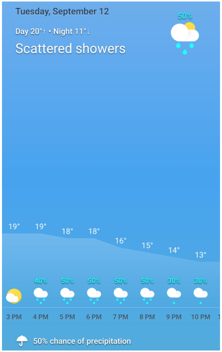
Committee: Uri Andrews, Jin-Yi Cai, **Paris Koutris**, Jignesh Patel, Jef Wijsen

Finding Consistent Answers from Inconsistent Data: Systems, Algorithms, and Complexity

JZ: want to go biking today at 6pm?

JZ: want to go biking today at 6pm?

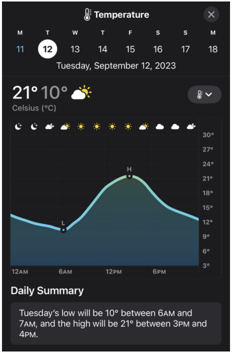
XO: . . . is that a good idea?

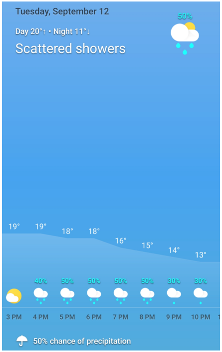


JZ: want to go biking today at 6pm?

XO: . . . is that a good idea?

JZ: that's not what I see . . .

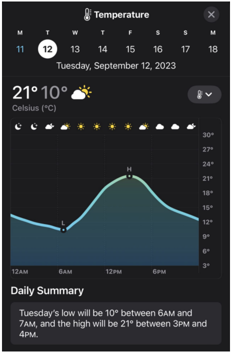


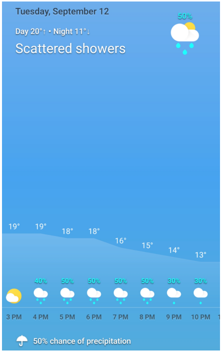


JZ: want to go biking today at 6pm?

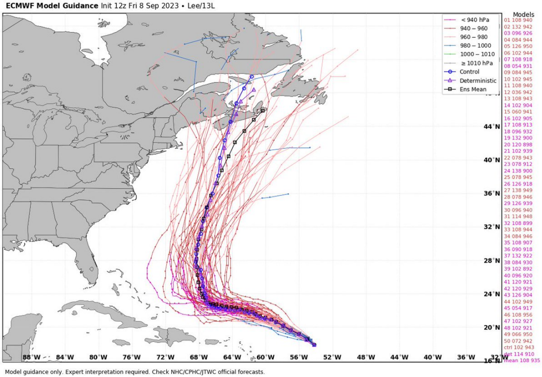
XO: . . . is that a good idea?

JZ: that's not what I see . . .

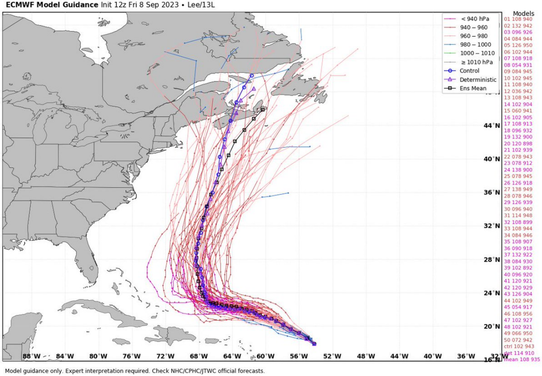




Us: let's play badminton instead . . .



- Alternatives from NLP, ML models ...
- Our focus: relational databases



- Alternatives from NLP, ML models ...
- Our focus: relational databases

Forecast

City	Weather
* MSN	Rainy
* MSN	Sunny
LA	Sunny
Seattle	Rainy

Activity

Weather	Biking	Badmin.
Rainy	No	Yes
Sunny	Yes	Yes
-37 deg.	No	No

- Inconsistent data: data that violates integrity constraints
- Primary key (PK) constraint: ≤ 1 tuple for each **PK value**

Forecast	
City	Weather
* MSN	Rainy
* MSN	Sunny
LA	Sunny
Seattle	Rainy

Activity		
Weather	Biking	Badmin.
Rainy	No	Yes
Sunny	Yes	Yes
-37 deg.	No	No

- Inconsistent data: data that violates integrity constraints
- Primary key (PK) constraint: ≤ 1 tuple for each **PK value**

Primary key constraint (violated)

- Metadata of `stackoverflow.com` as of 02/2021 from Stack Exchange Data Dump
- 551M rows, $\sim$400 GB

Table	# of rows	inconsistencyRatio	blockSize	# of Attributes
Users	14M	0%	1	14
Posts	53M	0%	1	20
PostHistory	141M	0.001%	4	9
Badges	40M	0.58%	941	4
Votes	213M	30.9%	1441	6

inconsistencyRatio = # facts violating PK constraint / # of rows

blockSize = max. # facts with the same PK

Finding Consistent Answers from Inconsistent Data: Systems, Algorithms, and Complexity

Finding consistent answers

Forecast

City	Weather
* MSN	Rainy
* MSN	Sunny
LA	Sunny
Seattle	Rainy

Activity

Weather	Biking	Badmin.
Rainy	No	Yes
Sunny	Yes	Yes
−37 deg.	No	No

q: find all cities that are suitable for badminton at 6pm

```
SELECT DISTINCT city
FROM Forecast, Activity
WHERE Forecast.weather = Activity.weather
  AND Badmin. = "Yes"
```

$q(x) = \exists y, z : \text{Forecast}(x, y) \wedge \text{Activity}(y, z, \texttt{"Yes"})$

$$
\begin{aligned}
q(\mathbf{db}) &= \{\text{Answers of } q \text{ on } \mathbf{db}\} \\
&= \{\texttt{city} \mid q_{[x \to \texttt{city}]} \text{ is true on } \mathbf{db}\} \\
&= \{\texttt{city} \mid \mathbf{db} \models q_{[x \to \texttt{city}]}\} \\
&= \{\text{MSN, LA, Seattle}\}
\end{aligned}
$$

Finding consistent answers

Forecast

City	Weather
* MSN	Rainy
* MSN	Sunny
LA	Sunny
Seattle	Rainy

Activity

Weather	Biking	Badmin.
Rainy	No	Yes
Sunny	Yes	Yes
-37 deg.	No	No

q: find all cities that are suitable for badminton at 6pm

```
SELECT DISTINCT city
FROM Forecast, Activity
WHERE Forecast.weather = Activity.weather
  AND Badmin. = "Yes"
```

$q(x) = \exists y, z : \text{Forecast}(x, y) \wedge \text{Activity}(y, z, \texttt{"Yes"})$

$$\begin{aligned} q(\mathbf{db}) &= \{\text{Answers of } q \text{ on } \mathbf{db}\} \\ &= \{\texttt{city} \mid q_{[x \to \texttt{city}]} \text{ is true on } \mathbf{db}\} \\ &= \{\texttt{city} \mid \mathbf{db} \models q_{[x \to \texttt{city}]}\} \\ &= \{\text{MSN, LA, Seattle}\} \end{aligned}$$

Finding consistent answers

Forecast

City	Weather
* MSN	Rainy
* MSN	Sunny
LA	Sunny
Seattle	Rainy

Activity

Weather	Biking	Badmin.
Rainy	No	Yes
Sunny	Yes	Yes
-37 deg.	No	No

q: find all cities that are suitable for badminton at 6pm

```
SELECT DISTINCT city
FROM Forecast, Activity
WHERE Forecast.weather = Activity.weather
  AND Badmin. = "Yes"
```

$$q(x) = \exists y, z : \text{Forecast}(x, y) \wedge \text{Activity}(y, z, \texttt{"Yes"})$$

$$\begin{aligned} q(\mathbf{db}) &= \{\text{Answers of } q \text{ on } \mathbf{db}\} \\ &= \{\texttt{city} \mid q_{[x \to \texttt{city}]} \text{ is true on } \mathbf{db}\} \\ &= \{\texttt{city} \mid \mathbf{db} \models q_{[x \to \texttt{city}]}\} \\ &= \{\text{MSN, LA, Seattle}\} \end{aligned}$$

Finding consistent answers

Forecast

City	Weather
* MSN	Rainy
* MSN	Sunny
LA	Sunny
Seattle	Rainy

Activity

Weather	Biking	Badmin.
Rainy	No	Yes
Sunny	Yes	Yes
−37 deg.	No	No

q: find all cities that are suitable for badminton at 6pm

```
SELECT DISTINCT city
FROM Forecast, Activity
WHERE Forecast.weather = Activity.weather
  AND Badmin. = "Yes"
```

$q(x) = \exists y, z : \text{Forecast}(x, y) \wedge \text{Activity}(y, z, \texttt{"Yes"})$

$$
\begin{aligned}
q(\mathbf{db}) &= \{\text{Answers of } q \text{ on } \mathbf{db}\} \\
&= \{\texttt{city} \mid q_{[x \to \texttt{city}]} \text{ is true on } \mathbf{db}\} \\
&= \{\texttt{city} \mid \mathbf{db} \models q_{[x \to \texttt{city}]}\} \\
&= \{\text{MSN, LA, Seattle}\}
\end{aligned}
$$

Finding consistent answers

Forecast

City	Weather
* MSN	Rainy
* MSN	Sunny
LA	Sunny
Seattle	Rainy

Activity

Weather	Biking	Badmin.
Rainy	No	Yes
Sunny	Yes	Yes
−37 deg.	No	No

q: find all cities that are suitable for badminton at 6pm

```
SELECT DISTINCT city
FROM Forecast, Activity
WHERE Forecast.weather = Activity.weather
  AND Badmin. = "Yes"
```

$$q(x) = \exists y, z : \text{Forecast}(x, y) \wedge \text{Activity}(y, z, \texttt{"Yes"})$$

$$\begin{aligned} q(\mathbf{db}) &= \{\text{Answers of } q \text{ on } \mathbf{db}\} \\ &= \{\texttt{city} \mid q_{[x \to \texttt{city}]} \text{ is true on } \mathbf{db}\} \\ &= \{\texttt{city} \mid \mathbf{db} \models q_{[x \to \texttt{city}]}\} \\ &= \{\text{MSN, LA, Seattle}\} \end{aligned}$$

So that we are on the same page...

DB system	DB theory	Logic
Database	Finite relations	Finite structure w/o func.
SQL Query w/o Aggr.	Query	First-order formula
Sel.-Proj.-Join Query	Conjunctive query (CQ)	Formula in $\mathbf{FO}(\exists, \wedge)$

Finding consistent answers

Forecast

City	Weather
* MSN	Rainy
* MSN	Sunny
LA	Sunny
Seattle	Rainy

Activity

Weather	Biking	Badmin.
Rainy	No	Yes
Sunny	Yes	Yes
−37 deg.	No	No

q: find all cities that are suitable for badminton at 6pm

$$q(\mathbf{db}) = \{\text{MSN, LA, Seattle}\} \ldots \text{ on dirty data}$$

Data cleaning

Finding consistent answers

Forecast

City	Weather
* MSN	Rainy
* MSN	Sunny
LA	Sunny
Seattle	Rainy

Activity

Weather	Biking	Badmin.
Rainy	No	Yes
Sunny	Yes	Yes
-37 deg.	No	No

q: find all cities that are suitable for badminton at 6pm

$$q(\mathbf{db}) = \{\text{MSN, LA, Seattle}\} \ldots \text{ on dirty data}$$

Data cleaning

Finding consistent answers

Forecast

City	Weather
* MSN	Rainy
* MSN	Sunny
LA	Sunny
Seattle	Rainy

Activity

Weather	Biking	Badmin.
Rainy	No	Yes
Sunny	Yes	Yes
−37 deg.	No	No

q: find all cities that are suitable for badminton at 6pm

$$q(\mathbf{db}) = \{\text{MSN, LA, Seattle}\} \ldots \text{ on dirty data}$$

Data cleaning

$$q(\mathbf{rep})$$

Finding consistent answers

Forecast

City	Weather
* MSN	Rainy
* MSN	Sunny
LA	Sunny
Seattle	Rainy

Activity

Weather	Biking	Badmin.
Rainy	No	Yes
Sunny	Yes	Yes
-37 deg.	No	No

q: find all cities that are suitable for badminton at 6pm

$$q(\mathbf{db}) = \{\text{MSN, LA, Seattle}\} \ldots \text{ on dirty data}$$

Data cleaning : 2 repairs

$$q(\mathbf{rep}) \qquad \text{vs.} \qquad q(\mathbf{rep}')$$

Finding consistent answers

Forecast

City	Weather
* MSN	Rainy
* MSN	Sunny
LA	Sunny
Seattle	Rainy

Activity

Weather	Biking	Badmin.
Rainy	No	Yes
Sunny	Yes	Yes
-37 deg.	No	No

q: find all cities that are suitable for badminton at 6pm

$$q(\mathbf{db}) = \{\text{MSN, LA, Seattle}\} \ldots \text{ on dirty data}$$

Data cleaning : 2 repairs (can be exponential...)

$q(\mathbf{rep})$ vs. $q(\mathbf{rep}')$

City	Weather
Chicago	Rainy/Sunny
Milwaukee	Rainy/Sunny
Oconomowoc	Rainy/Sunny
...	

Finding consistent answers

Forecast

City	Weather
* MSN	Rainy
* MSN	Sunny
LA	Sunny
Seattle	Rainy

Activity

Weather	Biking	Badmin.
Rainy	No	Yes
Sunny	Yes	Yes
−37 deg.	No	No

q: find all cities that are suitable for badminton at 6pm

$$q(\mathbf{db}) = \{\text{MSN, LA, Seattle}\} \ldots \text{ on dirty data}$$

Data cleaning : 2 repairs (can be exponential. . .)

$q(\mathbf{rep})$

Finding consistent answers

Forecast

City	Weather
* MSN	Rainy
* MSN	Sunny
LA	Sunny
Seattle	Rainy

Activity

Weather	Biking	Badmin.
Rainy	No	Yes
Sunny	Yes	Yes
−37 deg.	No	No

q: find all cities that are suitable for badminton at 6pm

$$q(\mathbf{db}) = \{\text{MSN, LA, Seattle}\} \ldots \text{ on dirty data}$$

Data cleaning : 2 repairs (can be exponential. . .)

Which answers are guaranteed to be returned on all repairs of dirty data?

$q(\mathbf{rep})$

Finding consistent answers

Forecast

City	Weather
* MSN	Rainy
* MSN	Sunny
LA	Sunny
Seattle	Rainy

Activity

Weather	Biking	Badmin.
Rainy	No	Yes
Sunny	Yes	Yes
−37 deg.	No	No

q: find all cities that are suitable for badminton at 6pm

$$q(\mathbf{db}) = \{\text{MSN, LA, Seattle}\} \ldots \text{ on dirty data}$$

Data cleaning : 2 repairs (can be exponential. . .)

$\boxed{\text{Which answers are guaranteed to be returned on } \underline{\text{all}} \text{ repairs of dirty data?}}$

$$\bigcap_{\mathbf{rep} \text{ is a repair of } \mathbf{db}} q(\mathbf{rep})$$

Finding consistent answers

Forecast

City	Weather
* MSN	Rainy
* MSN	Sunny
LA	Sunny
Seattle	Rainy

Activity

Weather	Biking	Badmin.
Rainy	No	Yes
Sunny	Yes	Yes
-37 deg.	No	No

q: find all cities that are suitable for badminton at 6pm

$$q(\mathbf{db}) = \{\text{MSN, LA, Seattle}\} \ldots \text{ on dirty data}$$

Data cleaning : 2 repairs (can be exponential. . .)

Which answers are guaranteed to be returned on <u>all</u> repairs of dirty data?

$$\bigcap_{\mathbf{rep} \text{ is a repair of } \mathbf{db}} q(\mathbf{rep}) = \{\textbf{MSN, LA, Seattle}\}$$

Finding consistent answers

Forecast

City	Weather
* MSN	Rainy
* MSN	Sunny
LA	Sunny
Seattle	Rainy

Activity

Weather	Biking	Badmin.
Rainy	No	Yes
Sunny	Yes	Yes
-37 deg.	No	No

q: find all cities that are suitable for badminton at 6pm

$$q(\mathbf{db}) = \{\text{MSN, LA, Seattle}\} \ldots \text{ on dirty data}$$

Data cleaning : 2 repairs (can be exponential...)

Which answers are guaranteed to be returned on <u>all</u> repairs of dirty data?

$$\text{Consistent Answer of } q \text{ over } \mathbf{db} = \bigcap_{\mathbf{rep} \text{ is a repair of } \mathbf{db}} q(\mathbf{rep}) = \{\textbf{MSN, LA, Seattle}\}$$

Finding consistent answers without enumeration

Forecast

City	Weather
* MSN	Rainy
* MSN	Sunny
LA	Sunny
Seattle	Rainy

Activity

Weather	Biking	Badmin.
Rainy	No	Yes
Sunny	Yes	Yes
-37 deg.	No	No

q: find all cities that are suitable for badminton at 6pm

q': find all cities that are suitable for badminton at 6pm

for all possible weather for the same city

```
SELECT DISTINCT city
FROM Forecast, Activity
WHERE Forecast.weather = Activity.weather
  AND (for all weather with the same Forecast.city,
      Badmin. = "Yes")
```

$$q'(x) = \exists y : \text{Forecast}(x, y) \wedge \forall y : (\text{Forecast}(x, y) \rightarrow \exists z : \text{Activity}(\underline{y}, z, \text{"Yes"}))$$

Finding consistent answers without enumeration

Forecast

City	Weather
* MSN	Rainy
* MSN	Sunny
LA	Sunny
Seattle	Rainy

Activity

Weather	Biking	Badmin.
Rainy	No	Yes
Sunny	Yes	Yes
−37 deg.	No	No

q: find all cities that are suitable for badminton at 6pm

q': find all cities that are suitable for badminton at 6pm

for all possible weather for the same city

```
SELECT DISTINCT city
FROM Forecast, Activity
WHERE Forecast.weather = Activity.weather
  AND (for all weather with the same Forecast.city,
      Badmin. = "Yes")
```

$$q'(x) = \exists y : \text{Forecast}(x, y) \wedge \forall y : (\text{Forecast}(x, y) \rightarrow \exists z : \text{Activity}(\underline{y}, z, \text{"Yes"}))$$

Finding consistent answers without enumeration

Forecast

City	Weather
* MSN	Rainy
* MSN	Sunny
LA	Sunny
Seattle	Rainy

Activity

Weather	Biking	Badmin.
Rainy	No	Yes
Sunny	Yes	Yes
−37 deg.	No	No

q: find all cities that are suitable for badminton at 6pm

q': find all cities that are suitable for badminton at 6pm

for all possible weather for the same city

```
SELECT DISTINCT city
FROM Forecast, Activity
WHERE Forecast.weather = Activity.weather
  AND (for all weather with the same Forecast.city,
      Badmin. = "Yes")
```

$$q'(x) = \exists y : \text{Forecast}(x, y) \wedge \forall y : (\text{Forecast}(x, y) \rightarrow \exists z : \text{Activity}(\underline{y}, z, \texttt{"Yes"}))$$

Finding consistent answers without enumeration

Forecast

City	Weather
* MSN	Rainy
* MSN	Sunny
LA	Sunny
Seattle	Rainy

Activity

Weather	Biking	Badmin.
Rainy	No	Yes
Sunny	Yes	Yes
−37 deg.	No	No

q: find all cities that are suitable for badminton at 6pm

q': find all cities that are suitable for badminton at 6pm

for all possible weather for the same city

Definition

q' is a first-order (**FO**) rewriting of q if

$$q'(\mathbf{db}) = \text{Consistent Answer of } q \text{ over } \mathbf{db} = \bigcap_{\mathbf{rep} \text{ is a repair of } \mathbf{db}} q(\mathbf{rep})$$

Not all q has an **FO**-rewriting. . .

Finding Consistent Answers from Inconsistent Data: Systems, Algorithms, and Complexity

For which queries can we find the consistent answers efficiently?

How efficient can we find the consistent answers?

Can we build a system finding the consistent answers?

Finding Consistent Answers from Inconsistent Data: Systems, Algorithms, and Complexity

For which queries can we find the consistent answers efficiently?

How efficient can we find the consistent answers?

Can we build a system finding the consistent answers?

Finding Consistent Answers from Inconsistent Data: Systems, Algorithms, and Complexity

For which queries can we find the consistent answers efficiently?

How efficient can we find the consistent answers?

Can we build a system finding the consistent answers?

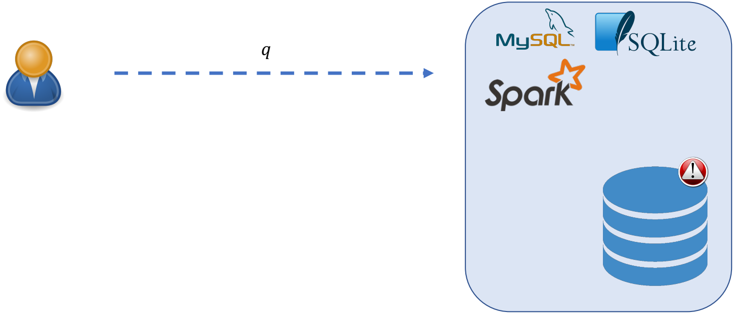
Finding Consistent Answers from Inconsistent Data: Systems, Algorithms, and Complexity

For which queries can we find the consistent answers efficiently?

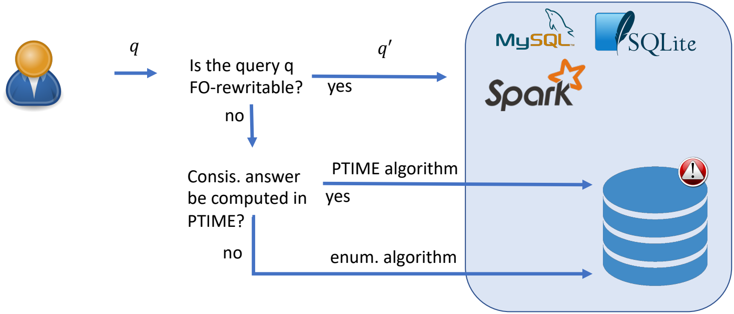
How efficient can we find the consistent answers?

Can we build a system finding the consistent answers?

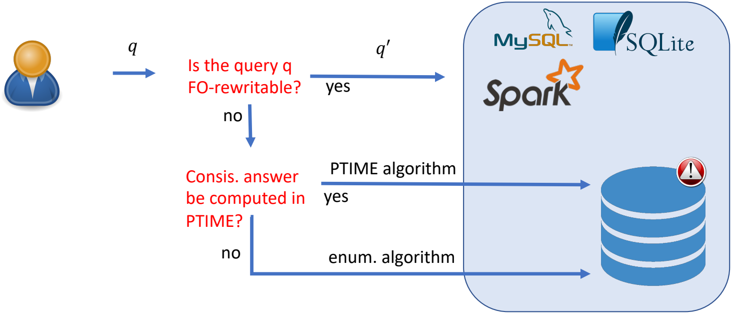
System motivations



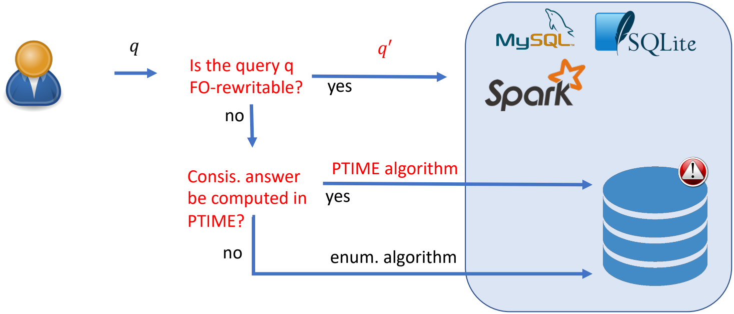
System motivations



System motivations



System motivations



Theoretical motivations

Problem: CERTAINTY(q), for a *fixed* query q as an **FO** sentence (`T/F`)

Input: a database **db** (as finite relations)

Question: does **rep** $\models q$ hold for every **rep** of **db** ?

Repair (**rep**): a maximal subset of **db** that satisfies the PK constraint

Theoretical motivations

Problem: CERTAINTY(q), for a *fixed* query q as an **FO** sentence ($\mathtt{T/F}$)

Input: a database **db** (as finite relations)

Question: does **rep** $\models q$ hold for every **rep** of **db** ?

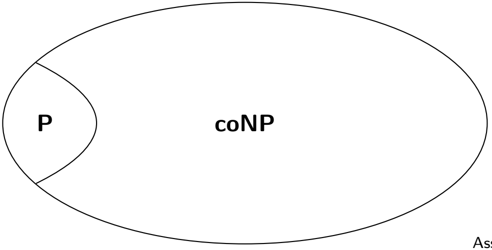
Repair (**rep**): a maximal subset of **db** that satisfies the PK constraint

Proposition

For every fixed query q, CERTAINTY(q) *is in* **coNP***.*

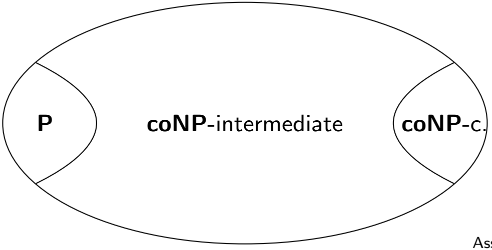
Proof: Guess a **rep** of **db** and check if **rep** $\models q$ in **P** (even in $\mathbf{AC}^0$) since q is fixed.

Theoretical motivations



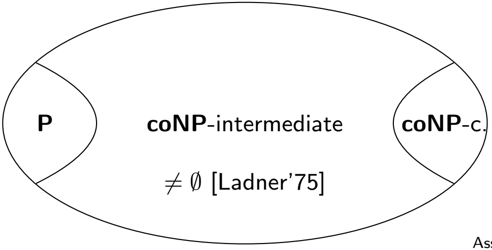
Assuming **P**$\neq$ **NP**. . .

Theoretical motivations



Assuming **P**$\neq$ **NP**...

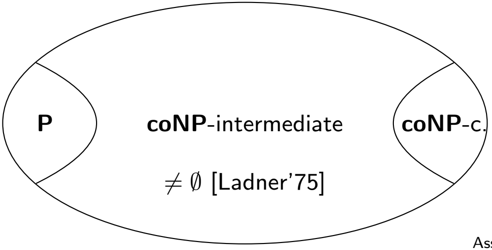
Theoretical motivations



Assuming **P**$\neq$ **NP**...

Possibly **NP**-intermediate: Graph Isomorphism, Factoring

Theoretical motivations



Assuming **P**$\neq$ **NP**...

Possibly **NP**-intermediate: Graph Isomorphism, Factoring

Conjecture

For every union of BCQ q, CERTAINTY(q) *is in* **P** *or* **coNP***-complete.*

unions of BCQ: $q_1 \vee \cdots \vee q_n$ for BCQs q_i in $\mathbf{FO}(\exists, \wedge)$

Relationship with Constraint Satisfaction Problems (CSP)

Conjecture

For every union of BCQ q, CERTAINTY(q) *is in* **P** *or* **coNP**-*complete.*

- Conservative CSP$\leq_p$ $\overline{\text{CERTAINTY}(q)}$ [Fontaine'15]
- CSP $\leq_p$ $\overline{\text{CQA}}$ for UCQs w.r.t. GAV constraints [Fontaine'15]

- Conservative CSP is in **P** or **NP**-complete. [Bulatov'03]
- CSP is in **P** or **NP**-complete. [Bulatov'17 & Zhuk'17]

Relationship with Constraint Satisfaction Problems (CSP)

Conjecture

For every union of BCQ q, $\text{CERTAINTY}(q)$ is in **P** *or* **coNP***-complete.*

- Conservative CSP$\leq_p \overline{\text{CERTAINTY}(q)}$ [Fontaine'15]
- CSP $\leq_p \overline{\text{CQA}}$ for UCQs w.r.t. GAV constraints [Fontaine'15]

- Conservative CSP is in **P** or **NP**-complete. [Bulatov'03]
- CSP is in **P** or **NP**-complete. [Bulatov'17 & Zhuk'17]

Our focus

Conjecture

*For every union of BCQ q, CERTAINTY(q) is in **P** or **coNP**-complete.*

Settled when q is self-join-free (SJF)! [Koutris & Wijsen, PODS'15, ICDT'19]

$$q(x) = \exists y, z : \text{Forecast}(x, y) \land \text{Activity}(y, z, \texttt{"Yes"}) \qquad \checkmark$$

$$q' = \exists y : \text{Flight}(\texttt{"Madison"}, y) \land \text{Flight}(y, \texttt{"LA"}) \qquad \times$$

$\mathcal{C}_{\text{forest}}$	α-acyclic	SJF two tables	SJF simple keys
FO	**FO**, non-**FO**	**P**, **coNP**-complete	**P**, **coNP**-complete
[FM, ICDT'05]	[Wijsen, PODS'10]	[KP, IPL'12]	[KS, ICDT'14]

theory

$\mathcal{C}_{\text{forest}}$	α-acyclic	SJF two tables	SJF simple keys	ConQuer
FO	**FO**, non-**FO**	**P**, **coNP**-complete	**P**, **coNP**-complete	$\mathcal{C}_{\text{forest}}$ via **FO**
[FM, ICDT'05]	[Wijsen, PODS'10]	[KP, IPL'12]	[KS, ICDT'14]	[FM, SIGMOD'05]

theory

system

SJF

FO, **L**-complete, **coNP**-complete

[KW, ICDT'19]

SJF

FO, **P**$\setminus$ **FO**, **coNP**-complete

[KW, PODS'15]

$\mathcal{C}_{\text{forest}}$

FO

[FM, ICDT'05]

α-acyclic

FO, non-**FO**

[Wijsen, PODS'10]

SJF two tables

P, **coNP**-complete

[KP, IPL'12]

SJF simple keys

P, **coNP**-complete

[KS, ICDT'14]

ConQuer

$\mathcal{C}_{\text{forest}}$ via **FO**

[FM, SIGMOD'05]

theory

system

SJF paths

FO, **NL**-complete, **P**-complete, **coNP**-complete

[KOW, PODS'21]

SJF

FO, **L**-complete, **coNP**-complete

[KW, ICDT'19]

SJF

FO, **P**$\setminus$ **FO**, **coNP**-complete

[KW, PODS'15]

$\mathcal{C}_{\text{forest}}$	α-acyclic	SJF two tables	SJF simple keys	ConQuer
FO	**FO**, non-**FO**	**P**, **coNP**-complete	**P**, **coNP**-complete	$\mathcal{C}_{\text{forest}}$ via **FO**
[FM, ICDT'05]	[Wijsen, PODS'10]	[KP, IPL'12]	[KS, ICDT'14]	[FM, SIGMOD'05]

theory

system

SJF rooted trees (and beyond)

FO, **P**$\setminus$ **FO**, **coNP**-complete

[KOW, PODS'24]

SJF paths

FO, **NL**-complete, **P**-complete, **coNP**-complete

[KOW, PODS'21]

SJF

FO, **L**-complete, **coNP**-complete

[KW, ICDT'19]

SJF

FO, **P**$\setminus$ **FO**, **coNP**-complete

[KW, PODS'15]

$\mathcal{C}_{\text{forest}}$	α-acyclic	SJF two tables	SJF simple keys	ConQuer
FO	**FO**, non-**FO**	**P**, **coNP**-complete	**P**, **coNP**-complete	$\mathcal{C}_{\text{forest}}$ via **FO**
[FM, ICDT'05]	[Wijsen, PODS'10]	[KP, IPL'12]	[KS, ICDT'14]	[FM, SIGMOD'05]

theory

system

SJF rooted trees (and beyond)

FO, **P** $\setminus$ **FO**, **coNP**-complete

[KOW, PODS'24]

SJF paths

FO, **NL**-complete, **P**-complete, **coNP**-complete

[KOW, PODS'21]

SJF

FO, **L**-complete, **coNP**-complete

[KW, ICDT'19]

SJF

FO, **P** $\setminus$ **FO**, **coNP**-complete

[KW, PODS'15]

Conquesto

SJF via **FO**

[AJLSW, CIKM'20]

$\mathcal{C}_{\text{forest}}$

FO

[FM, ICDT'05]

α-acyclic

FO, non-**FO**

[Wijsen, PODS'10]

SJF two tables

P, **coNP**-complete

[KP, IPL'12]

SJF simple keys

P, **coNP**-complete

[KS, ICDT'14]

ConQuer

$\mathcal{C}_{\text{forest}}$ via **FO**

[FM, SIGMOD'05]

theory

system

SJF rooted trees (and beyond)

FO, **P**$\setminus$ **FO**, **coNP**-complete

[KOW, PODS'24]

SJF paths

FO, **NL**-complete, **P**-complete, **coNP**-complete

[KOW, PODS'21]

SJF

FO, **L**-complete, **coNP**-complete

[KW, ICDT'19]

SJF

FO, **P**$\setminus$ **FO**, **coNP**-complete

[KW, PODS'15]

$\mathcal{C}_{\text{forest}}$

FO

[FM, ICDT'05]

α-acyclic

FO, non-**FO**

[Wijsen, PODS'10]

SJF two tables

P, **coNP**-complete

[KP, IPL'12]

SJF simple keys

P, **coNP**-complete

[KS, ICDT'14]

theory

CAvSAT

* via SAT

[DK, SAT'19, ICDE'21]

EQUIP

* via BIP

[KPT, VLDB'13]

Conquesto

SJF via **FO**

[AJLSW, CIKM'20]

ConQuer

$\mathcal{C}_{\text{forest}}$ via **FO**

[FM, SIGMOD'05]

system

SJF rooted trees (and beyond)

FO, **P** $\setminus$ **FO**, **coNP**-complete

[KOW, PODS'24]

SJF paths

FO, **NL**-complete, **P**-complete, **coNP**-complete

[KOW, PODS'21]

SJF

FO, **L**-complete, **coNP**-complete

[KW, ICDT'19]

SJF

FO, **P** $\setminus$ **FO**, **coNP**-complete

[KW, PODS'15]

$\mathcal{C}_{\text{forest}}$

FO

[FM, ICDT'05]

α-acyclic

FO, non-**FO**

[Wijsen, PODS'10]

SJF two tables

P, **coNP**-complete

[KP, IPL'12]

SJF simple keys

P, **coNP**-complete

[KS, ICDT'14]

theory

CAvSAT

* via SAT

[DK, SAT'19, ICDE'21]

EQUIP

* via BIP

[KPT, VLDB'13]

LinCQA

PPJT via **FO** in $O(N)$

[FKOW, SIGMOD'23]

Conquesto

SJF via **FO**

[AJLSW, CIKM'20]

ConQuer

$\mathcal{C}_{\text{forest}}$ via **FO**

[FM, SIGMOD'05]

system

SJF rooted trees (and beyond)

FO, **P**$\setminus$ **FO**, **coNP**-complete

[KOW, PODS'24]

SJF paths

FO, **NL**-complete, **P**-complete, **coNP**-complete

[KOW, PODS'21]

SJF

FO, **L**-complete, **coNP**-complete

[KW, ICDT'19]

SJF

FO, **P**$\setminus$ **FO**, **coNP**-complete

[KW, PODS'15]

$\mathcal{C}_{\text{forest}}$

FO

[FM, ICDT'05]

α-acyclic

FO, non-**FO**

[Wijsen, PODS'10]

SJF two tables

P, **coNP**-complete

[KP, IPL'12]

SJF simple keys

P, **coNP**-complete

[KS, ICDT'14]

theory

CAvSAT

* via SAT

[DK, SAT'19, ICDE'21]

EQUIP

* via BIP

[KPT, VLDB'13]

LinCQA

PPJT via **FO** in $O(N)$

[FKOW, SIGMOD'23]

Conquesto

SJF via **FO**

[AJLSW, CIKM'20]

ConQuer

$\mathcal{C}_{\text{forest}}$ via **FO**

[FM, SIGMOD'05]

system

It starts from *Acyclic Queries*...

Acyclic query evaluation

```
SELECT DISTINCT 1
FROM Forecast, Activity
WHERE Forecast.weather
      = Activity.weather
  AND Activity.Badmin = "Yes"
```

$$q = \exists x, y, z : \text{Forecast}(\underline{x}, y) \wedge \text{Activity}(\underline{y}, z, \texttt{"Yes"})$$

$\text{Forecast}(\underline{x}, y)$ — $\text{Activity}(\underline{y}, z, \texttt{"Yes"})$ Join Tree of q

Acyclic query evaluation

```
SELECT DISTINCT 1
FROM Forecast, Activity
WHERE Forecast.weather
      = Activity.weather
  AND Activity.Badmin = "Yes"
```

$$q = \exists x, y, z : \text{Forecast}(\underline{x}, y) \wedge \text{Activity}(\underline{y}, z, \texttt{"Yes"})$$

$\boxed{\text{Forecast}(\underline{x}, y)}$ — $\boxed{\text{Activity}(\underline{y}, z, \texttt{"Yes"})}$ Join Tree of q

Acyclic query evaluation

```
SELECT DISTINCT 1
FROM Forecast, Activity
WHERE Forecast.weather
      = Activity.weather
  AND Activity.Badmin = "Yes"
```

$$q = \exists x, y, z : \text{Forecast}(\underline{x}, y) \wedge \text{Activity}(\underline{y}, z, \texttt{"Yes"})$$

$\boxed{\text{Forecast}(\underline{x}, y)}$ — $\boxed{\text{Activity}(\underline{y}, z, \texttt{"Yes"})}$ Join Tree of q

Yannakakis [VLDB'81]

The answer to every **Boolean** acyclic query can be computed in $O(|\mathbf{db}|)$.

Yannakakis [VLDB'81] Our result

consistent answer

The ~~answer~~ to every **Boolean** acyclic query can be computed in $O(|\mathbf{db}|)$.

$\wedge$

with a pair-pruning join tree (PPJT)

Yannakakis [VLDB'81] Our result

consistent answer

The ~~answer~~ to every **Boolean** acyclic query can be computed in $O(|\mathbf{db}|)$.

$\wedge$

with a pair-pruning join tree (PPJT)

non-Boolean $\leq_T^P$ **Boolean**

```sql
SELECT
  DISTINCT Posts.Id, Posts.Title
FROM
  Posts, PostHistory, Votes, Comments
WHERE
  Posts.Tags LIKE "%SQL%"
  AND Posts.id = PostHistory.PostId
  AND Posts.id = Comments.PostId
  AND Posts.id = Votes.PostId
  AND Votes.BountyAmount > 100
  AND PostHistory.PostHistoryTypeId = 2
  AND Comments.score = 0
```

```sql
WITH candidates AS (
  SELECT
    DISTINCT C.UserId, C.CreationDate, P.Id, P.Title
  FROM
    Posts P, PostHistory PH, Votes V, Comments C
  WHERE
    P.Tags LIKE "%SQL%"
    AND P.id = PH.PostId
    AND P.id = C.PostId
    AND P.id = V.PostId
    AND V.BountyAmount > 100
    AND PH.PostHistoryTypeId = 2
    AND C.score = 0
),

Posts_bad_key AS (
  SELECT P.Id
  FROM Posts P
  WHERE P.Tags not LIKE "%SQL%" OR P.Tags IS NULL

  UNION ALL

  SELECT Id
  FROM (
    SELECT distinct Id, Title
    FROM Posts
  )t
  GROUP BY Id
  HAVING count(*) > 1
),

Posts_good_join AS (
  SELECT P.Id, P.Title
  FROM Posts P
  WHERE NOT EXISTS (
    SELECT *
    FROM Posts_bad_key
    WHERE P.Id = Posts_bad_key.Id
  )
),

PostHistory_bad_key AS (
  SELECT PH.PostId, PH.CreationDate, PH.UserId,
      PH.PostHistoryTypeId
  FROM PostHistory PH
  WHERE PH.PostHistoryTypeId <> 2
),

PostHistory_good_join AS (
  SELECT PH.PostId
  FROM PostHistory PH
  WHERE NOT EXISTS (
    SELECT *
    FROM PostHistory_bad_key
    WHERE PH.PostId = PostHistory_bad_key.PostId AND
      PH.CreationDate = PostHistory_bad_key.CreationDate
          AND
    PH.UserId = PostHistory_bad_key.UserId AND
    PH.PostHistoryTypeId
      = PostHistory_bad_key.PostHistoryTypeId
  )
),

Votes_bad_key AS (
  SELECT V.PostId, V.UserId, V.CreationDate
  FROM Votes V
  WHERE V.BountyAmount <= 100 or V.BountyAmount IS null
),

Votes_good_join AS (
  SELECT V.PostId
  FROM Votes V
  WHERE NOT EXISTS (
    SELECT *
    FROM Votes_bad_key
    WHERE
    V.PostId = Votes_bad_key.PostId AND
    V.UserId = Votes_bad_key.UserId AND
    V.CreationDate = Votes_bad_key.CreationDate
  )
),

Comments_bad_key AS (
  SELECT C.CreationDate, C.UserId, candidates.Title
  FROM Comments C
  JOIN candidates ON (
    C.CreationDate = candidates.CreationDate
    AND C.UserId = candidates.UserId)
  WHERE C.score <> 0

  UNION ALL

  SELECT C.CreationDate, C.UserId, candidates.Title
  FROM Comments C
  JOIN candidates ON (
    C.CreationDate = candidates.CreationDate
    AND C.UserId = candidates.UserId)
  LEFT OUTER JOIN Posts_good_join ON (
    C.PostId = Posts_good_join.Id
    AND candidates.Title = Posts_good_join.Title)
  LEFT OUTER JOIN PostHistory_good_join ON (
    C.PostId = PostHistory_good_join.PostId)
  LEFT OUTER JOIN Votes_good_join ON (
    C.PostId = Votes_good_join.PostId)
  WHERE (
    Posts_good_join.Id IS NULL
    OR PostHistory_good_join.PostId IS NULL
    OR Votes_good_join.PostId IS NULL
    OR Posts_good_join.Title IS NULL
  )
),

Comments_good_join AS (
  SELECT candidates.Id, candidates.Title
```

Original query (prev. slide) + primary key info $\xrightarrow{\text{LinCQA}}$ Query rewriting

```sql
WITH candidates AS (
  SELECT
    DISTINCT C.UserId, C.CreationDate, P.Id, P.Title
  FROM
    Posts P, PostHistory PH, Votes V, Comments C
  WHERE
    P.Tags LIKE "%SQL%"
    AND P.id = PH.PostId
    AND P.id = C.PostId
    AND P.id = V.PostId
    AND V.BountyAmount > 100
    AND PH.PostHistoryTypeId = 2
    AND C.score = 0
),

Posts_bad_key AS (
  SELECT P.Id
  FROM Posts P
  WHERE P.Tags not LIKE "%SQL%" OR P.Tags IS NULL

  UNION ALL

  SELECT Id
  FROM (
    SELECT distinct Id, Title
    FROM Posts
  )t
  GROUP BY Id
  HAVING count(*) > 1
),

Posts_good_join AS (
  SELECT P.Id, P.Title
  FROM Posts P
  WHERE NOT EXISTS (
    SELECT *
    FROM Posts_bad_key
    WHERE P.Id = Posts_bad_key.Id
  )
),

PostHistory_bad_key AS (
  SELECT PH.PostId, PH.CreationDate, PH.UserId,
         PH.PostHistoryTypeId
  FROM PostHistory PH
  WHERE PH.PostHistoryTypeId <> 2
),

PostHistory_good_join AS (
  SELECT PH.PostId
  FROM PostHistory PH
  WHERE NOT EXISTS (
    SELECT *
    FROM PostHistory_bad_key
    WHERE PH.PostId = PostHistory_bad_key.PostId AND
      PH.CreationDate = PostHistory_bad_key.CreationDate
          AND
      PH.UserId = PostHistory_bad_key.UserId AND
      PH.PostHistoryTypeId
        = PostHistory_bad_key.PostHistoryTypeId
  )
),

Votes_bad_key AS (
  SELECT V.PostId, V.UserId, V.CreationDate
  FROM Votes V
  WHERE V.BountyAmount <= 100 or V.BountyAmount IS null
),

Votes_good_join AS (
  SELECT V.PostId
  FROM Votes V
  WHERE NOT EXISTS (
    SELECT *
    FROM Votes_bad_key
    WHERE
    V.PostId = Votes_bad_key.PostId AND
    V.UserId = Votes_bad_key.UserId AND
    V.CreationDate = Votes_bad_key.CreationDate
  )
),

Comments_bad_key AS (
  SELECT C.CreationDate, C.UserId, candidates.Title
  FROM Comments C
  JOIN candidates ON (
    C.CreationDate = candidates.CreationDate
    AND C.UserId = candidates.UserId)
  WHERE C.score <> 0

  UNION ALL

  SELECT C.CreationDate, C.UserId, candidates.Title
  FROM Comments C
  JOIN candidates ON (
    C.CreationDate = candidates.CreationDate
    AND C.UserId = candidates.UserId)
  LEFT OUTER JOIN Posts_good_join ON (
    C.PostId = Posts_good_join.Id
    AND candidates.Title = Posts_good_join.Title)
  LEFT OUTER JOIN PostHistory_good_join ON (
    C.PostId = PostHistory_good_join.PostId)
  LEFT OUTER JOIN Votes_good_join ON (
    C.PostId = Votes_good_join.PostId)
  WHERE (
    Posts_good_join.Id IS NULL
    OR PostHistory_good_join.PostId IS NULL
    OR Votes_good_join.PostId IS NULL
    OR Posts_good_join.Title IS NULL
  )
),

Comments_good_join AS (
  SELECT candidates.Id, candidates.Title
```

Original query (prev. slide) + primary key info $\xrightarrow{\text{LinCQA}}$ Query rewriting

PPJT is a wide class

- $+$ $\subset$ **S**election, **P**rojection, **J**oin queries
- $+$ star/snowflake schema (e.g. 14/21 TPC-H)
- $+$ Every acyclic query in $\mathcal{C}_{\text{forest}}$ [Fuxman & Miller'05] has a PPJT

- $-$ no self-joins...
- $-$ no aggregation (yet) [Dixit & Kolaitis, 2022] [El Khalfioui & Wijsen, 2022]

PPJT is a wide class

+ $\subset$ **S**election, **P**rojection, **J**oin queries
+ star/snowflake schema (e.g. 14/21 TPC-H)
+ Every acyclic query in $\mathcal{C}_{\text{forest}}$ [Fuxman & Miller'05] has a PPJT

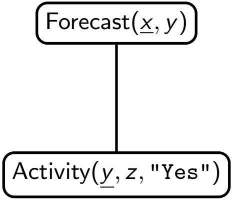
− no self-joins. . .
− no aggregation (yet) [Dixit & Kolaitis, 2022] [El Khalfioui & Wijsen, 2022]

From *Join Tree* to Pair-pruning Join Tree (PPJT)

Pair-pruning join tree (PPJT)

A join tree rooted at some atom is a PPJT if

$\boxed{\text{the root of every subtree is } \underline{\text{unattacked}} \text{ in the subtree}}$



Pair-pruning join tree (PPJT)

A join tree rooted at some atom is a PPJT if

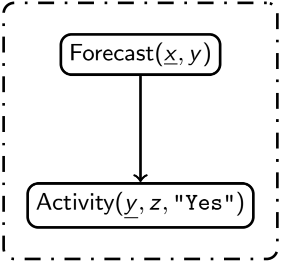
$$\boxed{\text{the root of every subtree is } \underline{\text{unattacked}} \text{ in the subtree}}$$

$$\begin{tikzcd} \boxed{\text{Forecast}(\underline{x}, y)} \arrow[d] \\ \boxed{\text{Activity}(\underline{y}, z, \texttt{"Yes"})} \end{tikzcd}$$

Pair-pruning join tree (PPJT)

A join tree rooted at some atom is a PPJT if

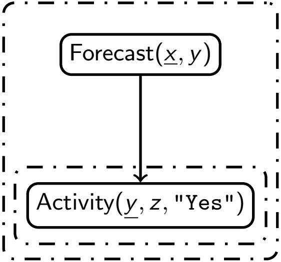
$\boxed{\text{the root of every subtree is } \underline{\text{unattacked}} \text{ in the subtree}}$



Pair-pruning join tree (PPJT)

A join tree rooted at some atom is a PPJT if

$\boxed{\text{the root of every subtree is } \underline{\text{unattacked}} \text{ in the subtree}}$



LinCQA: From PPJT to **FO**-rewriting

Remove a primary key if some tuple with this primary key is "bad"

$\text{Forecast}(\underline{x}, y)$

$\downarrow$

$\text{Activity}(\underline{y}, z, \texttt{"Yes"})$

$\text{Forecast}_{join}() \text{ :- } \text{Forecast}(x, y), \neg\text{Forecast}_{fkey}(x)$

$\text{Forecast}_{fkey}(x) \text{ :- } \text{Forecast}(x, y), \neg\text{Activity}_{join}(y)$

$\boxed{\forall\text{Child} : \text{Root}_{fkey}(\vec{x}) \text{ :- } \text{Root}(\underline{\vec{x}}, \vec{y}), \neg\text{Child}_{join}(\vec{\alpha})}$

$\boxed{\text{Child}_{join}(\vec{\alpha}) \text{ :- } \text{Child}(\underline{\vec{u}}, \vec{v}), \neg\text{Child}_{fkey}(\vec{u})}$

$\text{Activity}_{join}(y) \text{ :- } \text{Activity}(y, z, w), \neg\text{Activity}_{fkey}(y)$

$\text{Activity}_{fkey}(y) \text{ :- } \text{Activity}(y, z, w), w \neq \texttt{"Yes"}$

also expressible in SQL!

runs in $O(N)$

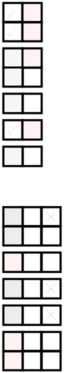
LinCQA: From PPJT to **FO**-rewriting

Remove a primary key if some tuple with this primary key is "bad"

$\text{Forecast}(\underline{x}, y)$

$\downarrow$

$\text{Activity}(\underline{y}, z, \texttt{"Yes"})$



$\text{Forecast}_{join}() \text{ :- } \text{Forecast}(x, y), \neg\text{Forecast}_{fkey}(x)$

$\text{Forecast}_{fkey}(x) \text{ :- } \text{Forecast}(x, y), \neg\text{Activity}_{join}(y)$

$\forall\text{Child} : \text{Root}_{fkey}(\vec{x}) \text{ :- } \text{Root}(\underline{\vec{x}}, \vec{y}), \neg\text{Child}_{join}(\vec{\alpha})$

$\text{Child}_{join}(\vec{\alpha}) \text{ :- } \text{Child}(\underline{\vec{u}}, \vec{v}), \neg\text{Child}_{fkey}(\vec{u})$

$\text{Activity}_{join}(y) \text{ :- } \text{Activity}(y, z, w), \neg\text{Activity}_{fkey}(y)$

$\text{Activity}_{fkey}(y) \text{ :- } \text{Activity}(y, z, w), w \neq \texttt{"Yes"}$

also expressible in SQL!
runs in $O(N)$

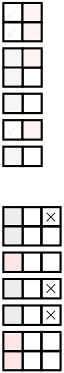
LinCQA: From PPJT to **FO**-rewriting

Remove a primary key if some tuple with this primary key is "bad"

$\text{Forecast}(\underline{x}, y)$

$\downarrow$

$\text{Activity}(\underline{y}, z, \texttt{"Yes"})$



$\text{Forecast}_{join}()$:- $\text{Forecast}(x, y), \neg\text{Forecast}_{fkey}(x)$

$\text{Forecast}_{fkey}(x)$:- $\text{Forecast}(x, y), \neg\text{Activity}_{join}(y)$

$\forall\text{Child} : \text{Root}_{fkey}(\vec{x})$:- $\text{Root}(\vec{x}, \vec{y}), \neg\text{Child}_{join}(\vec{\alpha})$

$\text{Child}_{join}(\vec{\alpha})$:- $\text{Child}(\vec{u}, \vec{v}), \neg\text{Child}_{fkey}(\vec{u})$

$\text{Activity}_{join}(y)$:- $\text{Activity}(y, z, w), \neg\text{Activity}_{fkey}(y)$

$\text{Activity}_{fkey}(y)$:- $\text{Activity}(y, z, w), w \neq \texttt{"Yes"}$

also expressible in SQL!

runs in $O(N)$

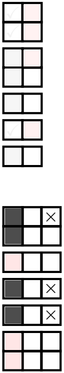
LinCQA: From PPJT to **FO**-rewriting

Remove a primary key if some tuple with this primary key is "bad"

$\text{Forecast}(\underline{x}, y)$

$\downarrow$

$\text{Activity}(\underline{y}, z, \texttt{"Yes"})$



$\text{Forecast}_{join}() \text{ :- } \text{Forecast}(x, y), \neg\text{Forecast}_{fkey}(x)$

$\text{Forecast}_{fkey}(x) \text{ :- } \text{Forecast}(x, y), \neg\text{Activity}_{join}(y)$

$\boxed{\forall\text{Child} : \text{Root}_{fkey}(\vec{x}) \text{ :- } \text{Root}(\underline{\vec{x}}, \vec{y}), \neg\text{Child}_{join}(\vec{\alpha})}$

$\boxed{\text{Child}_{join}(\vec{\alpha}) \text{ :- } \text{Child}(\underline{\vec{u}}, \vec{v}), \neg\text{Child}_{fkey}(\vec{u})}$

$\text{Activity}_{join}(y) \text{ :- } \text{Activity}(y, z, w), \neg\text{Activity}_{fkey}(y)$

$\text{Activity}_{fkey}(y) \text{ :- } \text{Activity}(y, z, w), w \neq \texttt{"Yes"}$

also expressible in SQL!

runs in $O(N)$

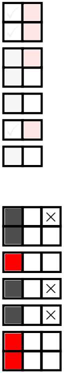
LinCQA: From PPJT to **FO**-rewriting

Remove a primary key if some tuple with this primary key is "bad"

$\text{Forecast}(\underline{x}, {\color{red}y})$

$\downarrow$

$\text{Activity}(\underline{{\color{red}y}}, z, \texttt{"Yes"})$



$\text{Forecast}_{join}() \text{ :- } \text{Forecast}(x, y), \neg\text{Forecast}_{fkey}(x)$

$\text{Forecast}_{fkey}(x) \text{ :- } \text{Forecast}(x, y), \neg\text{Activity}_{join}({\color{red}y})$

$\forall\text{Child} : \text{Root}_{fkey}(\vec{x}) \text{ :- } \text{Root}(\underline{\vec{x}}, \vec{y}), \neg\text{Child}_{join}(\vec{\alpha})$

$\text{Child}_{join}(\vec{\alpha}) \text{ :- } \text{Child}(\underline{\vec{u}}, \vec{v}), \neg\text{Child}_{fkey}(\vec{u})$

$\text{Activity}_{join}({\color{red}y}) \text{ :- } \text{Activity}(y, z, w), \neg\text{Activity}_{fkey}(y)$

$\text{Activity}_{fkey}(y) \text{ :- } \text{Activity}(y, z, w), w \neq \texttt{"Yes"}$

also expressible in SQL!
runs in $O(N)$

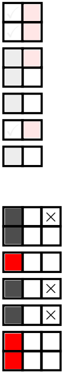
LinCQA: From PPJT to **FO**-rewriting

Remove a primary key if some tuple with this primary key is "bad"

$\text{Forecast}(\underline{x}, {\color{red}y})$

$\downarrow$

$\text{Activity}(\underline{{\color{red}y}}, z, \texttt{"Yes"})$



$\text{Forecast}_{join}() \text{ :- } \text{Forecast}(x, y), \neg\text{Forecast}_{fkey}(x)$

$\text{Forecast}_{fkey}(x) \text{ :- } \text{Forecast}(x, y), \neg\text{Activity}_{join}({\color{red}y})$

$\boxed{\forall\text{Child} : \text{Root}_{fkey}(\vec{x}) \text{ :- } \text{Root}(\underline{\vec{x}}, \vec{y}), \neg\text{Child}_{join}({\color{red}\vec{\alpha}})}$

$\boxed{\text{Child}_{join}({\color{red}\vec{\alpha}}) \text{ :- } \text{Child}(\underline{\vec{u}}, \vec{v}), \neg\text{Child}_{fkey}(\vec{u})}$

$\text{Activity}_{join}({\color{red}y}) \text{ :- } \text{Activity}(y, z, w), \neg\text{Activity}_{fkey}(y)$

$\text{Activity}_{fkey}(y) \text{ :- } \text{Activity}(y, z, w), w \neq \texttt{"Yes"}$

also expressible in SQL!
runs in $O(N)$

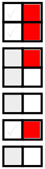
LinCQA: From PPJT to **FO**-rewriting

Remove a primary key if some tuple with this primary key is "bad"

$\text{Forecast}(\underline{x}, y)$

$\downarrow$

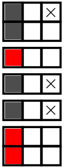
$\text{Activity}(\underline{y}, z, \texttt{"Yes"})$



$\text{Forecast}_{join}() \text{ :- } \text{Forecast}(x, y), \neg\text{Forecast}_{fkey}(x)$

$\text{Forecast}_{fkey}(x) \text{ :- } \text{Forecast}(x, y), \neg\text{Activity}_{join}(y)$

$\boxed{\forall\text{Child} : \text{Root}_{fkey}(\vec{x}) \text{ :- } \text{Root}(\underline{\vec{x}}, \vec{y}), \neg\text{Child}_{join}(\vec{\alpha})}$



$\boxed{\text{Child}_{join}(\vec{\alpha}) \text{ :- } \text{Child}(\underline{\vec{u}}, \vec{v}), \neg\text{Child}_{fkey}(\vec{u})}$

$\text{Activity}_{join}(y) \text{ :- } \text{Activity}(y, z, w), \neg\text{Activity}_{fkey}(y)$

$\text{Activity}_{fkey}(y) \text{ :- } \text{Activity}(y, z, w), w \neq \texttt{"Yes"}$

also expressible in SQL!

runs in $O(N)$

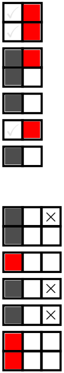
LinCQA: From PPJT to **FO**-rewriting

Remove a primary key if some tuple with this primary key is "bad"

$$\text{Forecast}(\underline{x}, y)$$

$$\downarrow$$

$$\text{Activity}(\underline{y}, z, \texttt{"Yes"})$$



$\text{Forecast}_{join}() \text{ :- } \text{Forecast}(x, y), \neg\text{Forecast}_{fkey}(x)$

$\text{Forecast}_{fkey}(x) \text{ :- } \text{Forecast}(x, y), \neg\text{Activity}_{join}(y)$

$\boxed{\forall \text{Child} : \text{Root}_{fkey}(\vec{x}) \text{ :- } \text{Root}(\underline{\vec{x}}, \vec{y}), \neg\text{Child}_{join}(\vec{\alpha})}$

$\boxed{\text{Child}_{join}(\vec{\alpha}) \text{ :- } \text{Child}(\underline{\vec{u}}, \vec{v}), \neg\text{Child}_{fkey}(\vec{u})}$

$\text{Activity}_{join}(y) \text{ :- } \text{Activity}(y, z, w), \neg\text{Activity}_{fkey}(y)$

$\text{Activity}_{fkey}(y) \text{ :- } \text{Activity}(y, z, w), w \neq \texttt{"Yes"}$

also expressible in SQL!

runs in $O(N)$

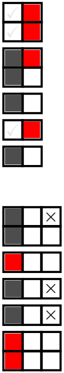
LinCQA: From PPJT to **FO**-rewriting

Remove a primary key if some tuple with this primary key is "bad"

$\text{Forecast}(\underline{x}, y)$

$\downarrow$

$\text{Activity}(\underline{y}, z, \texttt{"Yes"})$



$\text{Forecast}_{join}() \text{ :- } \text{Forecast}(x, y), \neg\text{Forecast}_{fkey}(x)$

$\text{Forecast}_{fkey}(x) \text{ :- } \text{Forecast}(x, y), \neg\text{Activity}_{join}(y)$

$\boxed{\forall\text{Child} : \text{Root}_{fkey}(\vec{x}) \text{ :- } \text{Root}(\underline{\vec{x}}, \vec{y}), \neg\text{Child}_{join}(\vec{\alpha})}$

$\boxed{\text{Child}_{join}(\vec{\alpha}) \text{ :- } \text{Child}(\underline{\vec{u}}, \vec{v}), \neg\text{Child}_{fkey}(\vec{u})}$

$\text{Activity}_{join}(y) \text{ :- } \text{Activity}(y, z, w), \neg\text{Activity}_{fkey}(y)$

$\text{Activity}_{fkey}(y) \text{ :- } \text{Activity}(y, z, w), w \neq \texttt{"Yes"}$

also expressible in SQL!
runs in $O(N)$

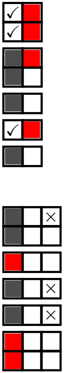
LinCQA: From PPJT to **FO**-rewriting

Remove a primary key if some tuple with this primary key is "bad"

$\text{Forecast}(\underline{x}, {\color{red}y})$

$\downarrow$

$\text{Activity}(\underline{{\color{red}y}}, z, \texttt{"Yes"})$



$\text{Forecast}_{join}() \text{ :- } \text{Forecast}(x, y), \neg\text{Forecast}_{fkey}(x)$

$\text{Forecast}_{fkey}(x) \text{ :- } \text{Forecast}(x, y), \neg\text{Activity}_{join}({\color{red}y})$

$\boxed{\forall\text{Child} : \text{Root}_{fkey}(\vec{x}) \text{ :- } \text{Root}(\underline{\vec{x}}, \vec{y}), \neg\text{Child}_{join}({\color{red}\vec{\alpha}})}$

$\boxed{\text{Child}_{join}({\color{red}\vec{\alpha}}) \text{ :- } \text{Child}(\underline{\vec{u}}, \vec{v}), \neg\text{Child}_{fkey}(\vec{u})}$

$\text{Activity}_{join}({\color{red}y}) \text{ :- } \text{Activity}(y, z, w), \neg\text{Activity}_{fkey}(y)$

$\text{Activity}_{fkey}(y) \text{ :- } \text{Activity}(y, z, w), w \neq \texttt{"Yes"}$

also expressible in SQL!
runs in $O(N)$

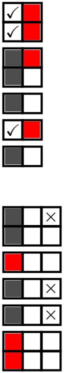
LinCQA: From PPJT to **FO**-rewriting

Remove a primary key if some tuple with this primary key is "bad"

Forecast($\underline{x}$, y)

$\downarrow$

Activity($\underline{y}$, z, "Yes")



$\text{Forecast}_{join}()$:- $\text{Forecast}(x, y), \neg\text{Forecast}_{fkey}(x)$

$\text{Forecast}_{fkey}(x)$:- $\text{Forecast}(x, y), \neg\text{Activity}_{join}(y)$

$\boxed{\forall\text{Child} : \text{Root}_{fkey}(\vec{x}) \text{ :- } \text{Root}(\underline{\vec{x}}, \vec{y}), \neg\text{Child}_{join}(\vec{\alpha})}$

$\boxed{\text{Child}_{join}(\vec{\alpha}) \text{ :- } \text{Child}(\underline{\vec{u}}, \vec{v}), \neg\text{Child}_{fkey}(\vec{u})}$

$\text{Activity}_{join}(y)$:- $\text{Activity}(y, z, w), \neg\text{Activity}_{fkey}(y)$

$\text{Activity}_{fkey}(y)$:- $\text{Activity}(y, z, w), w \neq$ "Yes"

also expressible in SQL!

runs in $O(N)$

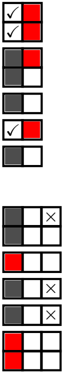
LinCQA: From PPJT to **FO**-rewriting

Remove a primary key if some tuple with this primary key is "bad"

$$\text{Forecast}(\underline{x}, {\color{red}y})$$

$$\downarrow$$

$$\text{Activity}(\underline{{\color{red}y}}, z, \texttt{"Yes"})$$



$\text{Forecast}_{join}()$:- $\text{Forecast}(x, y), \neg\text{Forecast}_{fkey}(x)$

$\text{Forecast}_{fkey}(x)$:- $\text{Forecast}(x, y), \neg\text{Activity}_{join}({\color{red}y})$

$\boxed{\forall\text{Child} : \text{Root}_{fkey}(\vec{x}) \text{ :- } \text{Root}(\underline{\vec{x}}, \vec{y}), \neg\text{Child}_{join}({\color{red}\vec{\alpha}})}$

$\boxed{\text{Child}_{join}({\color{red}\vec{\alpha}}) \text{ :- } \text{Child}(\underline{\vec{u}}, \vec{v}), \neg\text{Child}_{fkey}(\vec{u})}$

$\text{Activity}_{join}({\color{red}y})$:- $\text{Activity}(y, z, w), \neg\text{Activity}_{fkey}(y)$

$\text{Activity}_{fkey}(y)$:- $\text{Activity}(y, z, w), w \neq \texttt{"Yes"}$

also expressible in SQL!
runs in $O(N)$

From Boolean to non-Boolean

`SELECT DISTINCT A1, A2 FROM T WHERE A3 = 42`

Step 1 Evaluate directly

A1	A2
a	b
x	y
...	...

Step 2 Reduce to **Boolean** (using PPJT)

`SELECT DISTINCT 1 FROM T WHERE A3 = 42 AND A1 = `$\boxed{\texttt{a}}$` AND A2 = `$\boxed{\texttt{b}}$

$\boxed{\text{if } \textbf{yes}\text{, then output } (a, b)\text{, otherwise continue}}$

`SELECT DISTINCT 1 FROM T WHERE A3 = 42 AND A1 = `$\boxed{\texttt{x}}$` AND A2 = `$\boxed{\texttt{y}}$

...

$\xrightarrow{\text{LinCQA}}$ a single SQL/Datalog query

Acyclic q	PPJT	Yannakakis [VLDB'81]
Boolean q	$O(N)$	$O(N)$
non-Boolean q	$O(N \cdot \lvert \mathrm{OUT}_{\mathrm{inconsistent}} \rvert)$	$O(N \cdot \lvert \mathrm{OUT} \rvert)$
free-connex q	$O(N + \lvert \mathrm{OUT}_{\mathrm{consistent}} \rvert)$	$O(N + \lvert \mathrm{OUT} \rvert)$

Consistent answers of common join queries can be computed with no asymptotic overhead

Acyclic q	PPJT	Yannakakis [VLDB'81]
Boolean q	$O(N)$	$O(N)$
non-Boolean q	$O(N \cdot \lvert \mathrm{OUT}_{\mathrm{inconsistent}} \rvert)$	$O(N \cdot \lvert \mathrm{OUT} \rvert)$
free-connex q	$O(N + \lvert \mathrm{OUT}_{\mathrm{consistent}} \rvert)$	$O(N + \lvert \mathrm{OUT} \rvert)$

Consistent answers of common join queries can be computed with no asymptotic overhead

Experiments

Setup & Baselines

System	Target class	Interm. output	Backend
CAvSAT	*	SAT formula	SQL Server & MaxHS
Conquer	$\mathcal{C}_{\text{forest}}$	SQL	SQL Server
Improved Conquesto	SJF **FO**	SQL	SQL Server
LinCQA	PPJT	SQL	SQL Server

Stackoverflow data

- Metadata of `stackoverflow.com` as of 02/2021 from Stack Exchange Data Dump
- 551M rows, 400 GB

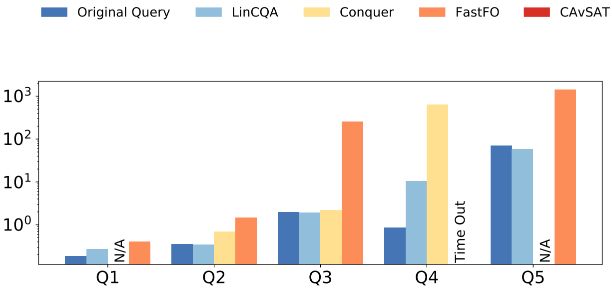
Table	# of rows	inconsistencyRatio	blockSize	# of Attributes
Users	14M	0%	1	14
Posts	53M	0%	1	20
PostHistory	141M	0.001%	4	9
Badges	40M	0.58%	941	4
Votes	213M	30.9%	1441	6

Stackoverflow results

Q_1 : Posts $\bowtie$ Votes $\quad Q_2$: Users $\bowtie$ Badges $\quad Q_3$: Users $\bowtie$ Posts

Q_4 : Users $\bowtie$ Posts $\bowtie$ Comments

Q_5 : Posts $\bowtie$ PostHistory $\bowtie$ Votes $\bowtie$ Comments



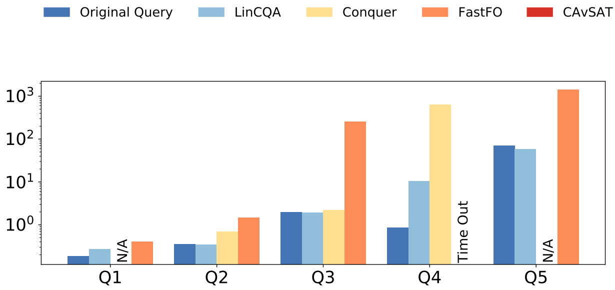
# poss.	27578	145	38320	3925	1250
# cons.	27578	145	38320	3925	1245

Stackoverflow results

Q_1 : Posts $\bowtie$ Votes $\quad Q_2$: Users $\bowtie$ Badges $\quad Q_3$: Users $\bowtie$ Posts

Q_4 : Users $\bowtie$ Posts $\bowtie$ Comments

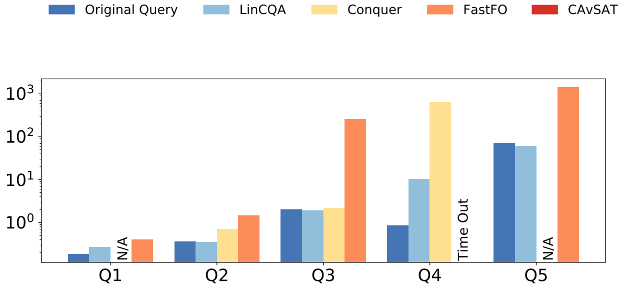
Q_5 : Posts $\bowtie$ PostHistory $\bowtie$ Votes $\bowtie$ Comments



# poss.	27578	145	38320	3925	1250
# cons.	27578	145	38320	3925	1245

Concluding remarks

Acyclic q	LinCQA [FKOW'23]	Yannakakis [VLDB'81]
Boolean q	$O(N)$	$O(N)$
non-Boolean q	$O(N \cdot \lvert \text{OUT}_{\text{inconsistent}} \rvert)$	$O(N \cdot \lvert \text{OUT} \rvert)$
free-connex q	$O(N + \lvert \text{OUT}_{\text{consistent}} \rvert)$	$O(N + \lvert \text{OUT} \rvert)$



	Q1	Q2	Q3	Q4	Q5
# poss.	27578	145	38320	3925	1250
# cons.	27578	145	38320	3925	1245

SJF rooted trees (and beyond)
FO, **P**$\setminus$ **FO**, **coNP**-complete
[KOW, PODS'24]

SJF paths
FO, **NL**-complete, **P**-complete, **coNP**-complete
[KOW, PODS'21]

SJF
FO, **L**-complete, **coNP**-complete
[KW, ICDT'19]

SJF
FO, **P**$\setminus$ **FO**, **coNP**-complete
[KW, PODS'15]

$\mathcal{C}_{\text{forest}}$
FO
[FM, ICDT'05]

α-acyclic
FO, non-**FO**
[Wijsen, PODS'10]

SJF two tables
P, **coNP**-complete
[KP, IPL'12]

SJF simple keys
P, **coNP**-complete
[KS, ICDT'14]

theory

CAvSAT
* via SAT
[DK, SAT'19, ICDE'21]

EQUIP
* via BIP
[KPT, VLDB'13]

LinCQA
PPJT via **FO** in $O(N)$
[FKOW, SIGMOD'23]

Conquesto
SJF via **FO**
[AJLSW, CIKM'20]

ConQuer
$\mathcal{C}_{\text{forest}}$ via **FO**
[FM, SIGMOD'05]

system

Why are self-joins complicated?

Problem: CERTAINTY(q), where

$$q = \exists x, y, z : \text{Forecast}(\underline{x}, y) \wedge \text{Activity}(\underline{y}, z, \texttt{"Yes"})$$

Input: a database **db** (as a finite set of relations)

Question: does **rep** $\models q$ hold for every **rep** of **db** ?

Forecast

City	Weather
* MSN	Rainy
* MSN	Sunny
LA	Sunny
Seattle	Rainy

Activity

Weather	Biking	Badmin.
Rainy	No	Yes
Sunny	Yes	Yes
-37 deg.	No	No

Forecast(MSN, Rainy) could *only* satisfy the predicate Forecast($\underline{x}$, y)

Problem: CERTAINTY(q), where

$$q = \exists x, y, z : \text{Forecast}(\underline{x}, y) \wedge \text{Activity}(\underline{y}, z, \texttt{"Yes"})$$

Input: a database **db** (as a finite set of relations)

Question: does **rep** $\models q$ hold for every **rep** of **db** ?

Forecast

City	Weather
* MSN	Rainy
* MSN	Sunny
LA	Sunny
Seattle	Rainy

Activity

Weather	Biking	Badmin.
Rainy	No	Yes
Sunny	Yes	Yes
−37 deg.	No	No

Forecast(MSN, Rainy) could *only* satisfy the predicate Forecast($\underline{x}$, y)

Problem: CERTAINTY(q), where

$$q = \exists x, y, z : R(\underline{x}, y) \wedge R(\underline{y}, z) \wedge X(\underline{z}, w) = RRX$$

Input: a database **db** (as a finite set of relations)

Question: does **rep** $\models q$ hold for every **rep** of **db** ?

R	$\underline{A_1}$	A_2
	0	1
	1	2
	1	3
	2	3

X	$\underline{B_1}$	B_2
	3	4

$R(1, 2)$ can satisfy either $R(\underline{x}, y)$ or $R(\underline{y}, z)$ now

rep$_1$

R RR RRX

rep$_2$

R RR

R RR RRX

Problem: CERTAINTY(q), where

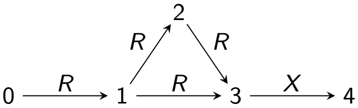
$$q = \exists x, y, z : R(\underline{x}, y) \wedge R(\underline{y}, z) \wedge X(\underline{z}, w) = RRX$$

Input: a database **db** (as a finite set of relations)

Question: does **rep** $\models q$ hold for every **rep** of **db** ?

R	$\underline{A_1}$	A_2
	0	1
	1	2
	1	3
	2	3

X	$\underline{B_1}$	B_2
	3	4



$R(1,2)$ can satisfy either $R(\underline{x}, y)$ or $R(\underline{y}, z)$ now

rep$_1$

RRX RRX RRX

rep$_2$

RRX RRX

RRX RRX RRX

Problem: CERTAINTY(q), where

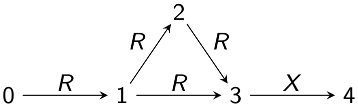
$$q = \exists x, y, z : R(\underline{x}, y) \wedge R(\underline{y}, z) \wedge X(\underline{z}, w) = RRX$$

Input: a database **db** (as a finite set of relations)

Question: does **rep** $\models q$ hold for every **rep** of **db** ?

R	$\underline{A_1}$	A_2
	0	1
	1	2
	1	3
	2	3

X	$\underline{B_1}$	B_2
	3	4



$R(1,2)$ can satisfy either $R(\underline{x}, y)$ or $R(\underline{y}, z)$ now

rep$_1$

rep$_2$

Problem: CERTAINTY(q), where

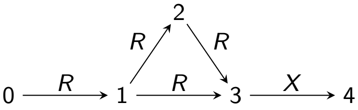
$$q = \exists x, y, z : R(\underline{x}, y) \land R(\underline{y}, z) \land X(\underline{z}, w) = RRX$$

Input: a database **db** (as a finite set of relations)

Question: does **rep** $\models q$ hold for every **rep** of **db** ?

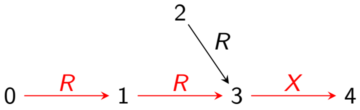
R	$\underline{A_1}$	A_2
	0	1
	1	2
	1	3
	2	3

X	$\underline{B_1}$	B_2
	3	4



$R(1, 2)$ can satisfy either $R(\underline{x}, y)$ or $R(\underline{y}, z)$ now

rep$_1$



RRX RRX RRX

rep$_2$

RRX RRX

RRX RRX RRX

Problem: CERTAINTY(q), where

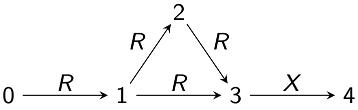
$$q = \exists x, y, z : R(\underline{x}, y) \wedge R(\underline{y}, z) \wedge X(\underline{z}, w) = RRX$$

Input: a database **db** (as a finite set of relations)

Question: does **rep** $\models q$ hold for every **rep** of **db** ?

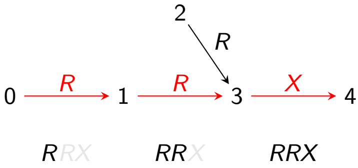
R	$\underline{A_1}$	A_2
	0	1
	1	2
	1	3
	2	3

X	$\underline{B_1}$	B_2
	3	4



$R(1,2)$ can satisfy either $R(\underline{x}, y)$ or $R(\underline{y}, z)$ now

rep$_1$



rep$_2$

Problem: CERTAINTY(q), where

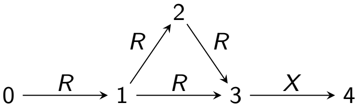
$$q = \exists x, y, z : R(\underline{x}, y) \wedge R(\underline{y}, z) \wedge X(\underline{z}, w) = RRX$$

Input: a database **db** (as a finite set of relations)

Question: does **rep** $\models q$ hold for every **rep** of **db** ?

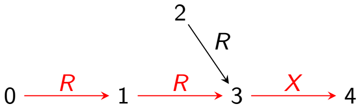
R	$\underline{A_1}$	A_2
	0	1
	1	2
	1	3
	2	3

X	$\underline{B_1}$	B_2
	3	4



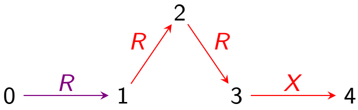
$R(1,2)$ can satisfy either $R(\underline{x}, y)$ or $R(\underline{y}, z)$ now

rep$_1$



RRX RRX RRX

rep$_2$



RRX RRX

RRX RRX RRX

Problem: CERTAINTY(q), where

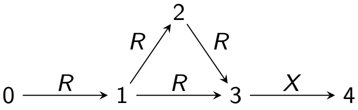
$$q = \exists x, y, z : R(\underline{x}, y) \wedge R(\underline{y}, z) \wedge X(\underline{z}, w) = RRX$$

Input: a database **db** (as a finite set of relations)

Question: does **rep** $\models q$ hold for every **rep** of **db** ?

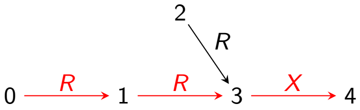
R	$\underline{A_1}$	A_2
	0	1
	1	2
	1	3
	2	3

X	$\underline{B_1}$	B_2
	3	4



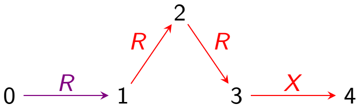
$R(1, 2)$ can satisfy either $R(\underline{x}, y)$ or $R(\underline{y}, z)$ now

rep$_1$



RRX RRX RRX

rep$_2$



RRX RRX

RRX RRX RRX

Problem: CERTAINTY(q), where

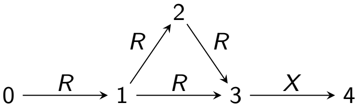
$$q = \exists x, y, z : R(\underline{x}, y) \wedge R(\underline{y}, z) \wedge X(\underline{z}, w) = RRX$$

Input: a database **db** (as a finite set of relations)

Question: does **rep** $\models q$ hold for every **rep** of **db** ?

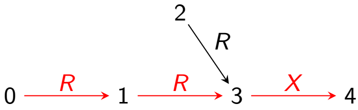
R	$\underline{A_1}$	A_2
	0	1
	1	2
	1	3
	2	3

X	$\underline{B_1}$	B_2
	3	4



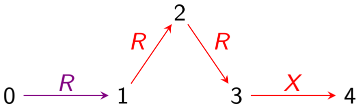
$R(1, 2)$ can satisfy either $R(\underline{x}, y)$ or $R(\underline{y}, z)$ now

rep$_1$



RRX RRX RRX

rep$_2$



RRX RRX

RRX RRX RRX

Problem: CERTAINTY(q), where

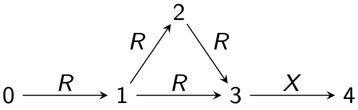
$$q = \exists x, y, z : R(\underline{x}, y) \wedge R(\underline{y}, z) \wedge X(\underline{z}, w) = RRX$$

Input: a database **db** (as a finite set of relations)

Question: does **rep** $\models q$ hold for every **rep** of **db** ?

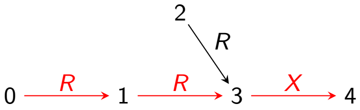
R	$\underline{A_1}$	A_2
	0	1
	1	2
	1	3
	2	3

X	$\underline{B_1}$	B_2
	3	4



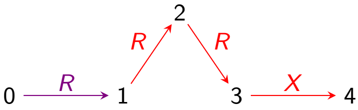
$R(1, 2)$ can satisfy either $R(\underline{x}, y)$ or $R(\underline{y}, z)$ now

rep$_1$



RRX RRX RRX

rep$_2$



RRX RRX

RRX RRX RRX

Problem: CERTAINTY(q), where

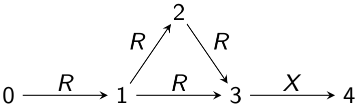
$$q = \exists x, y, z : R(\underline{x}, y) \land R(\underline{y}, z) \land X(\underline{z}, w) = RRX$$

Input: a database **db** (as a finite set of relations)

Question: does **rep** $\models q$ hold for every **rep** of **db** ?

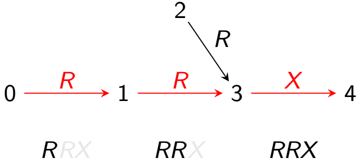
R	$\underline{A_1}$	A_2
	0	1
	1	2
	1	3
	2	3

X	$\underline{B_1}$	B_2
	3	4

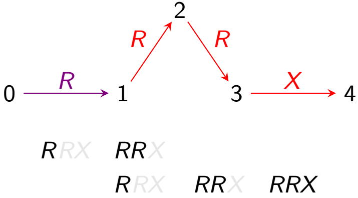


$R(1, 2)$ can satisfy either $R(\underline{x}, y)$ or $R(\underline{y}, z)$ now

rep$_1$



rep$_2$



Problem: CERTAINTY(q), where

$$q = \exists x, y, z : R(\underline{x}, y) \land R(\underline{y}, z) \land X(\underline{z}, w) = RRX$$

Input: a database **db** (as a finite set of relations)

Question: does **rep** $\models q$ hold for every **rep** of **db** ?

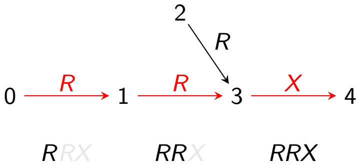
The key is to exploit this "rewinding" behavior

Proposition

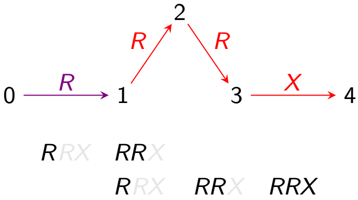
The following statements are equivalent:

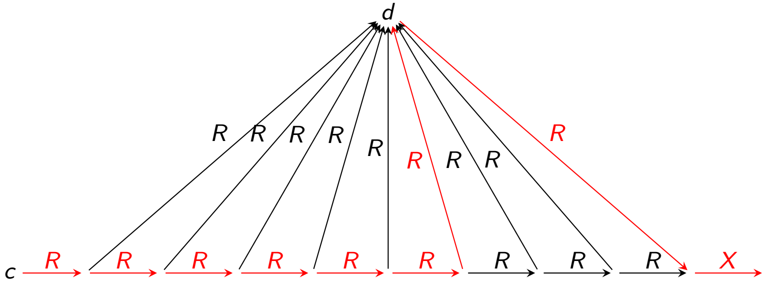
1. **db** *is a "yes"-instance for* CERTAINTY(RRX)*; and*
2. $\exists c$ *such that in all repairs, there exists a path of* $\underline{RR \cdot R^* \cdot X}$ *starting at* c.

rep$_1$



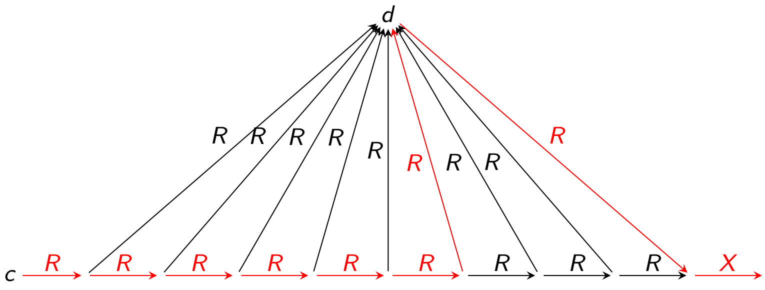
rep$_2$





"Reachability", "**NL**-complete"

How to find the regular expression?



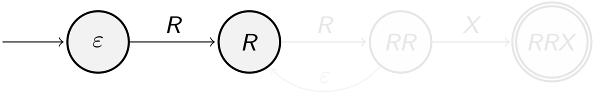
"Reachability", "**NL**-complete"

How to find the regular expression?

From path query to NFA

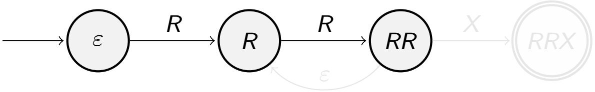
NFA(RRX) accepts RRR^*X

From path query to NFA



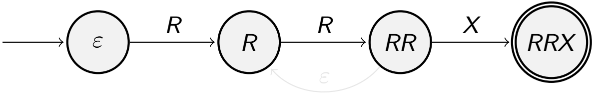
NFA(RRX) accepts RRR^*X

From path query to NFA



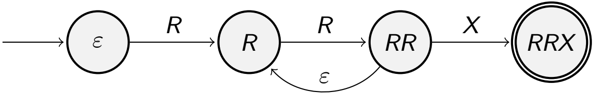
NFA(RRX) accepts RRR^*X

From path query to NFA



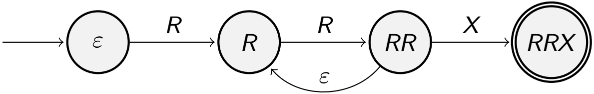
NFA(RRX) accepts RRR^*X

From path query to NFA



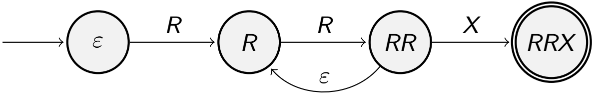
NFA(RRX) accepts RRR^*X

From path query to NFA



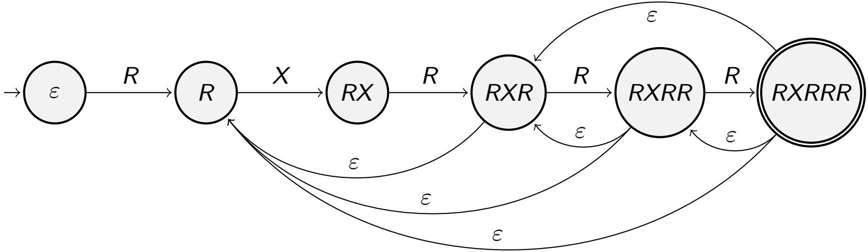
NFA(RRX) accepts RRR^*X

From path query to NFA



NFA(RRX) accepts RRR^*X

From path query to NFA (cont.)



NFA(*RXRRR*)

Path queries

$$
\begin{aligned}
q &= \exists x_0, x_1, \ldots, x_n : R_1(\underline{x_0}, x_1) \wedge R_2(\underline{x_1}, x_2) \wedge \cdots \wedge R_n(\underline{x_{n-1}}, x_n) \\
&= R_1 R_2 \ldots R_n
\end{aligned}
$$

- it can be that $R_i = R_j$ for $i \neq j$
- free variables & constants are easy extensions

Complexity classification for CERTAINTY(q)

NL-hard

$q_2 = RX\ RY$ $\qquad$ $RX\,\underline{RX\ RY} \in \text{NFA}(q_2)$

C$_1$: q is a prefix of every word in NFA(q)

$q_1 = RXRX$ $\qquad$ $\underline{RXRX}(RX)^* = \text{NFA}(q_1)$

FO-rewritable

Complexity classification for CERTAINTY(q)

coNP-complete

$q_4 = RXRX\ RYRY \qquad RXRXRYRXRYRY \in \text{NFA}(q_4)$

C$_2$: q is a <u>factor</u> of every word in NFA(q)

P

NL-hard

$q_2 = RX\ RY \qquad RX\underline{RX\ RY} \in \text{NFA}(q_2)$

C$_1$: q is a <u>prefix</u> of every word in NFA(q)

FO-rewritable

$q_1 = RXRX \qquad \underline{RXRX}(RX)^* = \text{NFA}(q_1)$

Complexity classification for CERTAINTY(q)

coNP-complete $q_4 = RXRX\ RYRY$ $RXRXRYRXRYRY \in \mathsf{NFA}(q_4)$

C$_2$: q is a factor of every word in NFA(q)

P

$q_3 = RX\ RYRY$

C$_{1.5}$: Whenever $q = uRvRw$, q is a factor of $uRvRvRw$; and whenever $q = uRv_1Rv_2Rw$ for consecutive occurrences of R, $v_1 = v_2$ or Rw is a prefix of Rv_1.

NL-hard

$q_2 = RX\ RY$ $RX\underline{RX\ RY} \in \mathsf{NFA}(q_2)$

C$_1$: q is a prefix of every word in NFA(q)

$q_1 = RXRX$ $\underline{RXRX}(RX)^* = \mathsf{NFA}(q_1)$

FO-rewritable

Complexity classification for CERTAINTY(q)

coNP-complete

$q_4 = RXRX\ RYRY$ $\quad$ $RXRXRYRXRYRY \in \text{NFA}(q_4)$

C$_2$: q is a factor of every word in NFA(q)

P-complete

$q_3 = RX\ RYRY$

C$_{1.5}$: Whenever $q = uRvRw$, q is a factor of $uRvRvRw$; and whenever $q = uRv_1Rv_2Rw$ for consecutive occurrences of R, $v_1 = v_2$ or Rw is a prefix of Rv_1.

NL-complete

$q_2 = RX\ RY$ $\quad$ $RX\underline{RX\ RY} \in \text{NFA}(q_2)$

C$_1$: q is a prefix of every word in NFA(q)

$q_1 = RXRX$ $\quad$ $\underline{RXRX}(RX)^* = \text{NFA}(q_1)$

FO-rewritable

C_1, $C_{1.5}$ and C_2 are decidable

$$C_1 : \quad q \text{ is a prefix of every word in } \mathsf{NFA}(q)$$
$$\iff \quad \text{Whenever } q = u \cdot \underline{Rv} \cdot Rw,\ q \text{ is a prefix of } u \cdot \underline{Rv} \cdot \underline{Rv} \cdot Rw.$$

$$C_2 : \quad q \text{ is a factor of every word in } \mathsf{NFA}(q)$$
$$\iff \quad \text{Whenever } q = u \cdot \underline{Rv} \cdot Rw,\ q \text{ is a factor of } u \cdot \underline{Rv} \cdot \underline{Rv} \cdot Rw.$$

Proposition

Let q be a path query satisfying C_2. The following statements are equivalent:

1. **db** is a "yes"-instance for CERTAINTY(q); and
2. $\exists c$ such that in all repairs, there exists a path accepted by NFA(q) starting in c.

C_2: q is a factor of every word in NFA(q)

When q satisfies C_1, $\mathsf{C}_{1.5}$, and C_2, item 2 can be checked in **FO**, **NL**, and **P** respectively

Proposition

Let q be a path query satisfying C_2. The following statements are equivalent:

1. **db** is a "yes"-instance for CERTAINTY(q); and
2. $\exists c$ such that in all repairs, there exists a path accepted by NFA(q) starting in c.

C_2: q is a factor of every word in NFA(q)

When q satisfies C_1, $C_{1.5}$, and C_2, item 2 can be checked in **FO**, **NL**, and **P** respectively

Proposition

Let q be a path query satisfying C_2. The following statements are equivalent:

1. **db** is a "yes"-instance for CERTAINTY(q); and
2. $\exists c$ such that in all repairs, there exists a path accepted by NFA(q) starting in c.

C_2: q is a factor of every word in NFA(q)

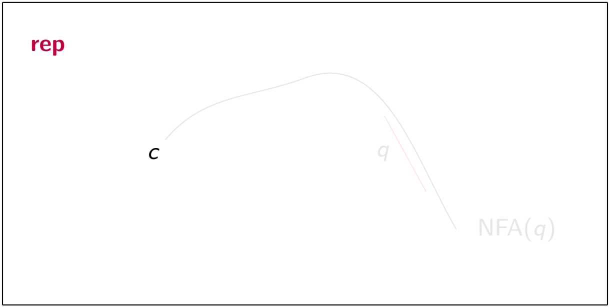
When q satisfies C_1, $C_{1.5}$, and C_2, item 2 can be checked in **FO**, **NL**, and **P** respectively

Proposition

Let q be a path query satisfying C_2. The following statements are equivalent:

1. **db** is a "yes"-instance for CERTAINTY(q); and
2. $\exists c$ such that in all repairs, there exists a path accepted by NFA(q) starting in c.

C_2: q is a factor of every word in NFA(q)



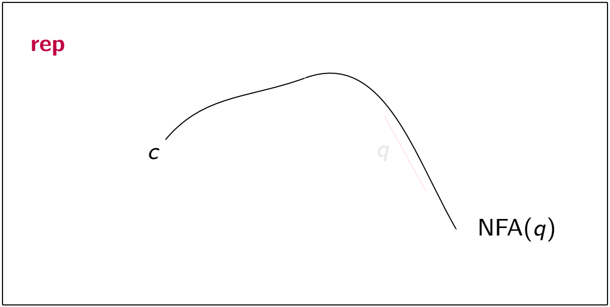
When q satisfies C_1, $C_{1.5}$, and C_2, item 2 can be checked in **FO**, **NL**, and **P** respectively

Proposition

Let q be a path query satisfying C_2. The following statements are equivalent:

1. **db** is a "yes"-instance for CERTAINTY(q); and
2. $\exists c$ such that in all repairs, there exists a path accepted by NFA(q) starting in c.

C_2: q is a factor of every word in NFA(q)



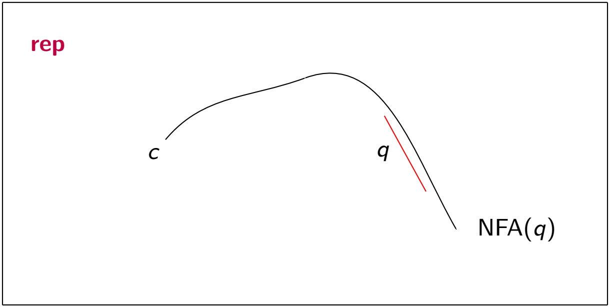
When q satisfies C_1, $C_{1.5}$, and C_2, item 2 can be checked in **FO**, **NL**, and **P** respectively

Proposition

Let q be a path query satisfying C_2. The following statements are equivalent:

1. **db** is a "yes"-instance for CERTAINTY(q); and
2. $\exists c$ such that in all repairs, there exists a path accepted by NFA(q) starting in c.

C_2: q is a factor of every word in NFA(q)



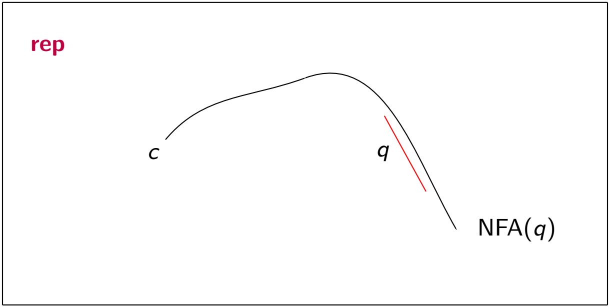
When q satisfies C_1, $C_{1.5}$, and C_2, item 2 can be checked in **FO**, **NL**, and **P** respectively

Proposition

Let q be a path query satisfying C_2. The following statements are equivalent:

1. **db** is a "yes"-instance for CERTAINTY(q); and
2. $\exists c$ such that in all repairs, there exists a path $\underline{\text{accepted by NFA}(q)}$ starting in c.

C_2: q is a factor of every word in NFA(q)



When q satisfies C_1, $C_{1.5}$, and C_2, item 2 can be checked in **FO**, **NL**, and **P** respectively

Hardness

Lemma

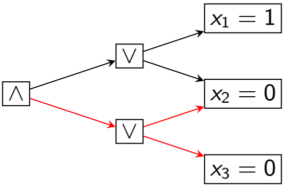
For a path query q, *via*

- *if q violates* C_1, *then* CERTAINTY(q) *is* **NL**-*hard;* *Reachability*
- *if q violates* $\mathsf{C}_{1.5}$, *then* CERTAINTY(q) *is* **P**-*hard;* *Monotone Circuit Value*
- *if q violates* C_2, *then* CERTAINTY(q) *is* **coNP**-*hard.* *Unsatisfiability*

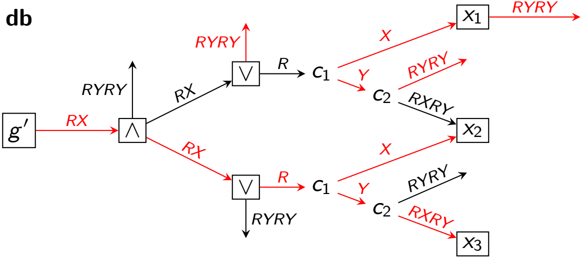
P-hardness

$q = RXRYRY$ violates $\mathsf{C}_{1.5}$

$\mathcal{C}$



db



The output of $\mathcal{C}$ is 0 iff **db** contains a falsifying repair

Complexity classification for Path Queries

coNP-complete $q_4 = RXRX\ RYRY$ $RXRXRYRXRYRY \in \text{NFA}(q_4)$

C_2: q is a <u>factor</u> of every word in $\text{NFA}(q)$

P-complete $q_3 = RX\ RYRY$

$\text{C}_{1.5}$: Whenever $q = uRvRw$, q is a factor of $uRvRvRw$; and whenever $q = uRv_1Rv_2Rw$ for consecutive occurrences of R, $v_1 = v_2$ or Rw is a prefix of Rv_1.

NL-complete $q_2 = RX\ RY$ $RX\underline{RX\ RY} \in \text{NFA}(q_2)$

C_1: q is a <u>prefix</u> of every word in $\text{NFA}(q)$

FO-rewritable $q_1 = RXRX$ $\underline{RXRX}(RX)^* = \text{NFA}(q_1)$

SJF rooted trees (and beyond)

FO, **P**$\setminus$ **FO**, **coNP**-complete

[KOW, PODS'24]

CAvSAT

* via SAT

[DK, SAT'19, ICDE'21]

SJF paths

FO, **NL**-complete, **P**-complete, **coNP**-complete

[KOW, PODS'21]

EQUIP

* via BIP

[KPT, VLDB'13]

SJF

FO, **L**-complete, **coNP**-complete

[KW, ICDT'19]

LinCQA

PPJT via **FO** in $O(N)$

[FKOW, SIGMOD'23]

SJF

FO, **P**$\setminus$ **FO**, **coNP**-complete

[KW, PODS'15]

Conquesto

SJF via **FO**

[AJLSW, CIKM'20]

$\mathcal{C}_{\text{forest}}$

FO

[FM, ICDT'05]

α-acyclic

FO, non-**FO**

[Wijsen, PODS'10]

SJF two tables

P, **coNP**-complete

[KP, IPL'12]

SJF simple keys

P, **coNP**-complete

[KS, ICDT'14]

ConQuer

$\mathcal{C}_{\text{forest}}$ via **FO**

[FM, SIGMOD'05]

theory

system

$$q = \exists x, y, z, w : R(\underline{x}, y) \wedge R(\underline{y}, z) \wedge X(\underline{z}, w) = RRX$$

$$q \text{ :- } R(\underline{x}, y), R(\underline{y}, z), X(\underline{z}, w)$$

$$\begin{array}{c} R \\ | \\ R \\ | \\ X \\ | \\ \perp \end{array} \qquad\qquad \begin{array}{c} x \\ | \\ y \\ | \\ z \\ | \\ w \end{array}$$

$$x \xrightarrow{R} y \xrightarrow{R} z \xrightarrow{X} w$$

no idea yet...

$$q = \exists x, y, z, w : R(\underline{x}, y) \wedge R(\underline{y}, z) \wedge X(\underline{z}, w) = RRX$$

$$q \text{ :- } R(\underline{x}, y), R(\underline{y}, z), X(\underline{z}, w)$$

$$\begin{tikzcd} R \arrow[d, dash] & & & x \arrow[d, dash] \\ R \arrow[d, dash] & & & y \arrow[d, dash] \\ X \arrow[d, dash] & & & z \arrow[d, dash] \\ \perp & & & w \end{tikzcd}$$

$$x \xrightarrow{R} y \xrightarrow{R} z \xrightarrow{X} w$$

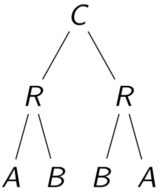
no idea yet...

$$q = \exists x, y, z, w : R(\underline{x}, y) \wedge R(\underline{y}, z) \wedge X(\underline{z}, w) = RRX$$
$$q \text{ :- } R(\underline{x}, y), R(\underline{y}, z), X(\underline{z}, w)$$

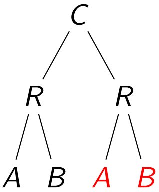
$$\begin{tikzcd} R \arrow[d, dash] \\ R \arrow[d, dash] \\ X \arrow[d, dash] \\ \perp \end{tikzcd} \qquad\qquad \begin{tikzcd} x \arrow[d, dash] \\ y \arrow[d, dash] \\ z \arrow[d, dash] \\ w \end{tikzcd}$$

$$x \xrightarrow{R} y \xrightarrow{R} z \xrightarrow{X} w$$

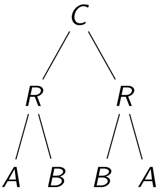
no idea yet. . .



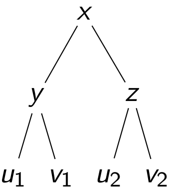
q_1



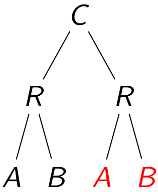
q_2



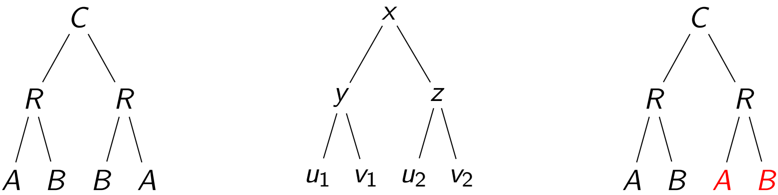
q_1



variable mapping

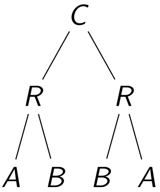


q_2

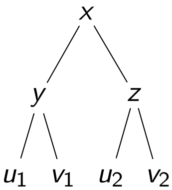


q_1 variable mapping q_2

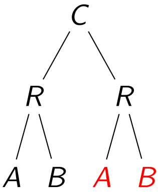
$$q_1 \text{ :- } C(\underline{x}, y, z), R(\underline{y}, u_1, v_1), A(\underline{u_1}), B(\underline{v_1}), R(\underline{z}, u_2, v_2), B(\underline{u_2}), A(\underline{v_2}).$$



q_1



variable mapping



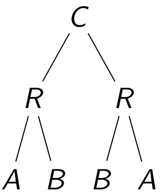
q_2

$$q_1 \text{ :- } C(\underline{x}, y, z), R(\underline{y}, u_1, v_1), A(\underline{u_1}), B(\underline{v_1}), R(\underline{z}, u_2, v_2), B(\underline{u_2}), A(\underline{v_2}).$$

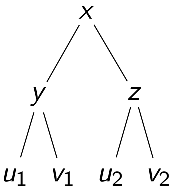
$$q_2 \text{ :- } C(\underline{x}, y, z), R(\underline{y}, u_1, v_1), A(\underline{u_1}), B(\underline{v_1}), R(\underline{z}, u_2, v_2), \color{red}{A(\underline{u_2}), B(\underline{v_2})}.$$

What about rewinding?

$$q = R \overbrace{RX}^{\text{replace}} \Longrightarrow R \overbrace{RRX}^{\text{with a ``previous'' word}}$$



q



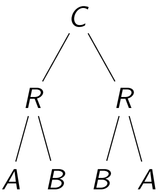
variable mapping

?

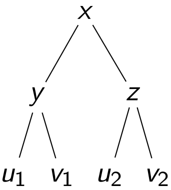
$q^{R: z \looparrowright y}$

What about rewinding?

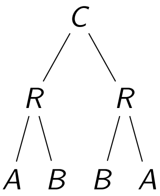
$$q = R\ \overbrace{RX}^{\text{replace}} \Longrightarrow R\ \overbrace{RRX}^{\text{with a ``previous'' word}}$$



q



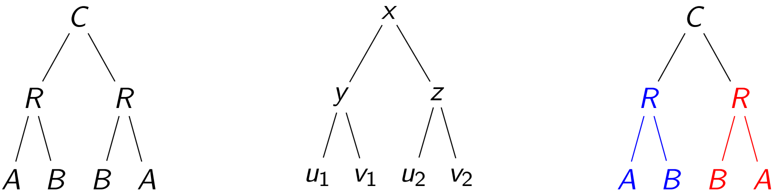
variable mapping



$q^{R:z \looparrowright y}$

What about rewinding?

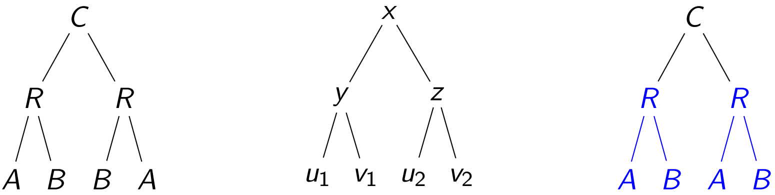
$$q = R \overbrace{RX}^{\text{replace}} \Longrightarrow R \overbrace{RRX}^{\text{with a ``previous'' word}}$$



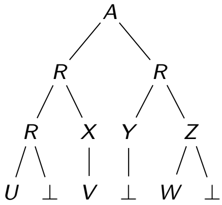
q variable mapping $q^{R: z \looparrowright y}$

What about rewinding?

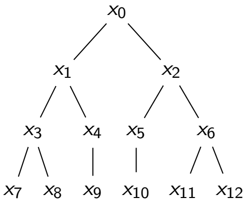
$$q = R \overbrace{RX}^{\text{replace}} \Longrightarrow R \overbrace{RRX}^{\text{with a ``previous'' word}}$$



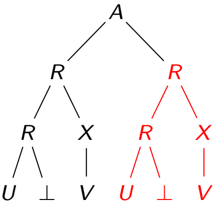
q variable mapping $q^{R:z \not\mapsto y}$



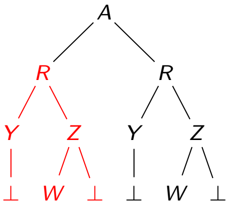
q



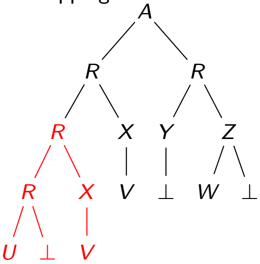
var. mapping



$q^{R:x_2 \hookrightarrow x_1}$



$q^{R:x_1 \hookrightarrow x_2}$



$q^{R:x_3 \hookrightarrow x_1}$

Classification on rooted trees

$C_2^{\clubsuit}$: for every $R\langle x\rangle$ and $R\langle y\rangle$ in q, there is a *homomorphism* from q to either

$$q^{R:x\nrightarrow y} \text{ or } q^{R:y\nrightarrow x}$$

q_1

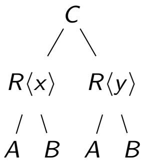
$q_1^{R:y\nrightarrow x} = q_1^{R:x\nrightarrow y}$

q_1 satisfies $C_2^{\clubsuit}$

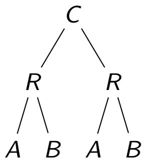
Classification on rooted trees

$C_2^{\clubsuit}$: for every $R\langle x\rangle$ and $R\langle y\rangle$ in q, there is a *homomorphism* from q to either

$$q^{R:x\looparrowright y} \text{ or } q^{R:y\looparrowright x}$$



q_1



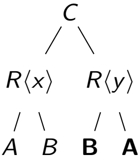
$${q_1}^{R:y\looparrowright x} = {q_1}^{R:x\looparrowright y}$$

q_1 satisfies $C_2^{\clubsuit}$

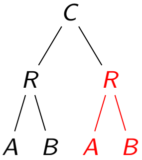
Classification on rooted trees

$C_2^\clubsuit$: for every $R\langle x \rangle$ and $R\langle y \rangle$ in q, there is a *homomorphism* from q to either

$$q^{R:x \looparrowright y} \text{ or } q^{R:y \looparrowright x}$$



q_2



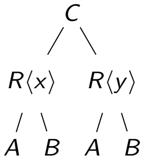
${q_2}^{R:y \looparrowright x}$

q_2 violates $C_2^\clubsuit$

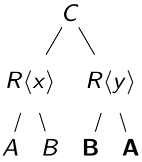
Classification on rooted trees

$C_2^\clubsuit$: for every $R\langle x\rangle$ and $R\langle y\rangle$ in q, there is a *homomorphism* from q to either

$$q^{R:x \hookrightarrow y} \text{ or } q^{R:y \hookrightarrow x}$$



q_1 satisfies $C_2^\clubsuit$



q_2 violates $C_2^\clubsuit$

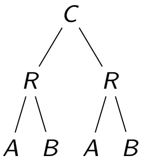
Theorem

If q satisfies $C_2^\clubsuit$, then CERTAINTY(q) *is in* **P**, *or otherwise* **coNP**-*complete.*

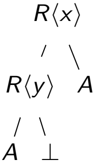
Classification on rooted trees

$C_1^{\clubsuit}$: for every $R\langle x\rangle$ and $R\langle y\rangle$ in q, there is a *root homomorphism* from q to either

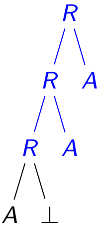
$$q^{R:x \looparrowright y} \text{ or } q^{R:y \looparrowright x}$$



q_1 satisfies $C_1^{\clubsuit}$



q_3 satisfies $C_1^{\clubsuit}$

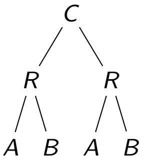


${q_3}^{R:y \looparrowright x}$

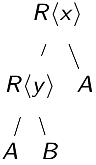
Classification on rooted trees

$C_1^{\clubsuit}$: for every $R\langle x\rangle$ and $R\langle y\rangle$ in q, there is a *root homomorphism* from q to either

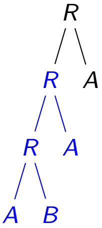
$$q^{R:x\looparrowright y} \text{ or } q^{R:y\looparrowright x}$$



q_1 satisfies $C_1^{\clubsuit}$



$q_4 : \neg C_1^{\clubsuit}, C_2^{\clubsuit}$



${q_4}^{R:y\looparrowright x}$

Theorem

If q satisfies $C_1^{\clubsuit}$, then CERTAINTY(q) *is in* **FO**, *or otherwise* **NL**-*hard.*

Rooted trees generalize paths

FO-rewritable

$C_1^{\clubsuit}$: for every $R\langle x\rangle$ and $R\langle y\rangle$ in q, there is a <u>*root homomorphism*</u> from q to either $q^{R:x \looparrowright y}$ or $q^{R:y \looparrowright x}$

Rooted trees generalize paths

NL-hard

FO-rewritable

$C_1^{\clubsuit}$: for every $R\langle x\rangle$ and $R\langle y\rangle$ in q, there is a *root homomorphism* from q to either $q^{R:x \looparrowright y}$ or $q^{R:y \looparrowright x}$

Rooted trees generalize paths

P

$\mathsf{C}_2^{\clubsuit}$: for every $R\langle x\rangle$ and $R\langle y\rangle$ in q, there is a <u>*homomorphism*</u> from q to either $q^{R:x \looparrowright y}$ or $q^{R:y \looparrowright x}$

NL-hard

FO-rewritable

$\mathsf{C}_1^{\clubsuit}$: for every $R\langle x\rangle$ and $R\langle y\rangle$ in q, there is a <u>*root homomorphism*</u> from q to either $q^{R:x \looparrowright y}$ or $q^{R:y \looparrowright x}$

Rooted trees generalize paths

coNP-complete

P

$C_2^\clubsuit$: for every $R\langle x\rangle$ and $R\langle y\rangle$ in q, there is a *homomorphism* from q to either $q^{R:x \nleftrightarrow y}$ or $q^{R:y \nleftrightarrow x}$

NL-hard

FO-rewritable

$C_1^\clubsuit$: for every $R\langle x\rangle$ and $R\langle y\rangle$ in q, there is a *root homomorphism* from q to either $q^{R:x \nleftrightarrow y}$ or $q^{R:y \nleftrightarrow x}$

Rooted trees generalize paths

coNP-complete

P

$C_2^{\clubsuit}$: for every $R\langle x\rangle$ and $R\langle y\rangle$ in q, there is a <u>*homomorphism*</u> from q to either $q^{R:x \nrightarrow y}$ or $q^{R:y \nrightarrow x}$

C_2: $q = u\ Rv\ Rw$ is a <u>factor</u> of $u\ Rv\ Rv\ Rw$

NL-hard

FO-rewritable

$C_1^{\clubsuit}$: for every $R\langle x\rangle$ and $R\langle y\rangle$ in q, there is a <u>*root homomorphism*</u> from q to either $q^{R:x \nrightarrow y}$ or $q^{R:y \nrightarrow x}$

C_1: $q = u\ Rv\ Rw$ is a <u>prefix</u> of $u\ Rv\ Rv\ Rw$

Good rooted trees are just “paths”

$C_2^{\clubsuit}$: for every $R\langle x\rangle$ and $R\langle y\rangle$ in q, there is a *homomorphism* from q to either

$$q^{R:x \hookrightarrow y} \text{ or } q^{R:y \hookrightarrow x}$$

Definition: $R\langle x\rangle \preceq_q R\langle y\rangle$ if

- $R\langle x\rangle$ is an ancestor of $R\langle y\rangle$ in q; or
- there is a homomorphism from q to $q^{R:y \hookrightarrow x}$

Proposition: If q satisfies $C_2^{\clubsuit}$, for every predicate name R, the relation $\preceq_q$ is a total preorder on all R-atoms.

$$R\langle x\rangle \quad \preceq_q \quad \begin{matrix} R\langle y\rangle \\ \\ R\langle z\rangle \end{matrix} \quad \cdots \quad \preceq_q \quad R\langle u\rangle$$

Good rooted trees are just "paths"

$C_2^{\clubsuit}$: for every $R\langle x\rangle$ and $R\langle y\rangle$ in q, there is a <u>*homomorphism*</u> from q to either

$$q^{R:x \nrightarrow y} \text{ or } q^{R:y \nrightarrow x}$$

Definition: $R\langle x\rangle \preceq_q R\langle y\rangle$ if

- $R\langle x\rangle$ is an ancestor of $R\langle y\rangle$ in q; or
- there is a homomorphism from q to $q^{R:y \nrightarrow x}$

Proposition: If q satisfies $C_2^{\clubsuit}$, for every predicate name R, the relation $\preceq_q$ is a total preorder on all R-atoms.

$$R\langle x\rangle \quad \preceq_q \quad \begin{matrix} R\langle y\rangle \\ \\ R\langle z\rangle \end{matrix} \quad \cdots \quad \preceq_q \quad R\langle u\rangle$$

For good trees, checking *one* repair is all you need

$C_2^{\clubsuit}$: for every $R\langle x \rangle$ and $R\langle y \rangle$ in q, there is a *homomorphism* from q to either

$$q^{R:x \looparrowright y} \text{ or } q^{R:y \looparrowright x}$$

Problem: CERTAINTY(q), for a rooted tree query q

Input: a database **db**

Question: does **rep** $\models q$ hold for every **rep** of **db** ?

rep$_1 \models q$? **rep**$_2 \models q$? **rep**$_3 \models q$? $\cdots$ **rep**$_{2^n} \models q$?

For good trees, checking *one* repair is all you need

$C_2^{\clubsuit}$: for every $R\langle x\rangle$ and $R\langle y\rangle$ in q, there is a <u>*homomorphism*</u> from q to either

$$q^{R:x \looparrowright y} \text{ or } q^{R:y \looparrowright x}$$

Problem:	CERTAINTY(q), for a rooted tree query q
Input:	a database **db**
Question:	does **rep** $\models q$ hold for every **rep** of **db** ?

$\textbf{rep}_1 \models q$? $\textbf{rep}_2 \models q$? $\textbf{rep}_3 \models q$? $\cdots$ $\textbf{rep}_{2^n} \models q$?

For good trees, checking *one* repair is all you need

$C_2^\clubsuit$: for every $R\langle x\rangle$ and $R\langle y\rangle$ in q, there is a <u>*homomorphism*</u> from q to either

$$q^{R:x \nrightarrow y} \text{ or } q^{R:y \nrightarrow x}$$

Problem: CERTAINTY(q), for a rooted tree query q

Input: a database **db**

Question: does **rep** $\models q$ hold for every **rep** of **db** ?

rep$_1 \models q$? **rep**$_2 \models q$? **rep**$_3 \models q$? $\cdots$ **rep*** **rep**$_{2^n} \models q$?

Proposition: If q satisfies $C_2^\clubsuit$, there exists some **rep*** of **db** that depends on q

For good trees, checking *one* repair is all you need

$C_2^{\clubsuit}$: for every $R\langle x\rangle$ and $R\langle y\rangle$ in q, there is a *homomorphism* from q to either

$$q^{R:x \nrightarrow y} \text{ or } q^{R:y \nrightarrow x}$$

Problem: CERTAINTY(q), for a rooted tree query q

Input: a database **db**

Question: does **rep** $\models q$ hold for every **rep** of **db** ?

$\mathbf{rep}_1 \models q$? $\mathbf{rep}_2 \models q$? $\mathbf{rep}_3 \models q$? $\cdots$ $\mathbf{rep}^*$ $\mathbf{rep}_{2^n} \models q$?

Proposition: If q satisfies $C_2^{\clubsuit}$, there exists some $\mathbf{rep}^*$ of **db** that depends on q such that

$$\mathbf{rep}^* \models q$$

For good trees, checking *one* repair is all you need

$\mathsf{C}_2^{\clubsuit}$: for every $R\langle x\rangle$ and $R\langle y\rangle$ in q, there is a *homomorphism* from q to either

$$q^{R:x \not\hookrightarrow y} \text{ or } q^{R:y \not\hookrightarrow x}$$

Problem: CERTAINTY(q), for a rooted tree query q

Input: a database **db**

Question: does **rep** $\models q$ hold for every **rep** of **db** ?

$\mathbf{rep}_1 \models q$? $\mathbf{rep}_2 \models q$? $\mathbf{rep}_3 \models q$? $\cdots$ $\mathbf{rep}^*$ $\mathbf{rep}_{2^n} \models q$?

Proposition: If q satisfies $\mathsf{C}_2^{\clubsuit}$, there exists some $\mathbf{rep}^*$ of **db** that depends on q such that

$$\mathbf{rep}^* \models q \iff \mathbf{rep} \models q \text{ for every } \mathbf{rep} \text{ of } \mathbf{db}.$$

For good trees, checking *one* repair is all you need

$C_2^{\clubsuit}$: for every $R\langle x\rangle$ and $R\langle y\rangle$ in q, there is a *homomorphism* from q to either

$$q^{R:x \nrightarrow y} \text{ or } q^{R:y \nrightarrow x}$$

Problem: CERTAINTY(q), for a rooted tree query q

Input: a database **db**

Question: does **rep** $\models q$ hold for every **rep** of **db** ?

$\mathbf{rep}_1 \models q$? $\mathbf{rep}_2 \models q$? $\mathbf{rep}_3 \models q$? $\cdots$ $\mathbf{rep}^*$ $\mathbf{rep}_{2^n} \models q$?

Proposition: If q satisfies $C_2^{\clubsuit}$, there exists some $\mathbf{rep}^*$ of **db** that depends on q such that

$$\mathbf{rep}^* \models q \quad \Longleftrightarrow \quad \mathbf{rep} \models q \text{ for every } \mathbf{rep} \text{ of } \mathbf{db}.$$

Moreover, one such $\mathbf{rep}^*$ can be found in **P**.

Initialization Step: **for every** $c \in \text{adom}(\mathbf{db})$ and leaf variable or constant u in q
add $\langle c, u\rangle$ to B **if** $u = c$ is a constant,
or the label of variable u in q is either $\perp$,
or L with $L(\underline{c}) \in \mathbf{db}$.

Iterative Rule: **for every** $c \in \text{adom}(\mathbf{db})$ and atom $R(\underline{y}, y_1, y_2, \dots, y_n)$ in q
add $\langle c, y\rangle$ to B **if** the following formula holds:

$$\exists \vec{d} : R(\underline{c}, \vec{d}) \in \mathbf{db} \wedge \forall \vec{d} : \left(R(\underline{c}, \vec{d}) \in \mathbf{db} \rightarrow \text{fact}(R(\underline{c}, \vec{d}), y) \right),$$

where

$$\text{fact}(R(\underline{c}, \vec{d}), y) = \underbrace{\left(\bigwedge_{1 \leq i \leq n} \langle d_i, y_i \rangle \in B \right)}_{\text{forward production}} \vee \underbrace{\left(\bigvee_{R[x] <_q R[y]} \text{fact}(R(\underline{c}, \vec{d}), x) \right)}_{\text{backward production}}$$

and $\vec{d} = \langle d_1, d_2, \dots, d_n \rangle$.

Classification for rooted trees

coNP-complete

P

C$_2^\clubsuit$: for every $R\langle x\rangle$ and $R\langle y\rangle$ in q, there is a <u>*homomorphism*</u> from q to either $q^{R:x \nrightarrow y}$ or $q^{R:y \nrightarrow x}$

C$_2$: $q = u\ Rv\ Rw$ is a <u>factor</u> of $u\ Rv\ Rv\ Rw$

NL-hard

FO-rewritable

C$_1^\clubsuit$: for every $R\langle x\rangle$ and $R\langle y\rangle$ in q, there is a <u>*root homomorphism*</u> from q to either $q^{R:x \nrightarrow y}$ or $q^{R:y \nrightarrow x}$

C$_1$: $q = u\ Rv\ Rw$ is a <u>prefix</u> of $u\ Rv\ Rv\ Rw$

Can be extended to "(Berge-acyclic) Graph queries" . . .

Classification for rooted trees

coNP-complete

P

$C_2^{\clubsuit}$: for every $R\langle x\rangle$ and $R\langle y\rangle$ in q, there is a <u>*homomorphism*</u> from q to either $q^{R:x \looparrowright y}$ or $q^{R:y \looparrowright x}$

C_2: $q = u\ Rv\ Rw$ is a <u>factor</u> of $u\ Rv\ Rv\ Rw$

NL-hard

FO-rewritable

$C_1^{\clubsuit}$: for every $R\langle x\rangle$ and $R\langle y\rangle$ in q, there is a <u>*root homomorphism*</u> from q to either $q^{R:x \looparrowright y}$ or $q^{R:y \looparrowright x}$

C_1: $q = u\ Rv\ Rw$ is a <u>prefix</u> of $u\ Rv\ Rv\ Rw$

Can be extended to "(Berge-acyclic) Graph queries" ...

Concluding remarks

Finding Consistent Answers from Inconsistent Data: Systems, Algorithms, and Complexity

kNN + missing values
[Karlaš et al., VLDB'21]

kNN + FD
P, **coNP**-complete
[FK, ICDT'22]

Bayes + missing
(BOFK, submitted)

ML + dirty data
...?

graph query
...**FO**, **P**$\setminus$ **FO**, **coNP**-complete?

SJF + (intgrty. const.)
...**FO**, non-**FO**?

rooted trees (and beyond)
FO, **P**$\setminus$ **FO**, **coNP**-complete
[KOW, PODS'24]

SJF + PK & (unary)FK
FO, non-**FO**
[HW, PODS'22]

path
FO, **NL**-complete, **P**-complete, **coNP**-complete
[KOW, PODS'21]

LinCQA$^+$
...SJF acyclic **FO** in $O(N)$?

SJF + multiple keys
FO, non-**FO**
[KW, PODS'20]

SJF
FO, **L**-complete, **coNP**-complete
[KW, ICDT'19]

LinCQA
PPJT via **FO** in $O(N)$
[FKOW, SIGMOD'23]

SJF + $\neg$
FO, non-**FO**
[KW, PODS'18]

Conjecture: For every (union of) BCQ q, CERTAINTY(q) is in **P** or **coNP**-complete.

Finding Consistent Answers from Inconsistent Data: Systems, Algorithms, and Complexity

kNN + missing values
[Karlaš et al., VLDB'21]

kNN + FD
P, **coNP**-complete
[FK, ICDT'22]

Bayes + missing
(BOFK, submitted)

ML + dirty data
. . . ?

graph query
. . . **FO**, **P**\ **FO**, **coNP**-complete?

SJF + (intgrty. const.)
. . . **FO**, non-**FO**?

rooted trees (and beyond)
FO, **P**\ **FO**, **coNP**-complete
[KOW, PODS'24]

SJF + PK & (unary)FK
FO, non-**FO**
[HW, PODS'22]

path
FO, **NL**-complete, **P**-complete, **coNP**-complete
[KOW, PODS'21]

LinCQA$^+$
. . . SJF acyclic **FO** in $O(N)$?

SJF + multiple keys
FO, non-**FO**
[KW, PODS'20]

SJF
FO, **L**-complete, **coNP**-complete
[KW, ICDT'19]

LinCQA
PPJT via **FO** in $O(N)$
[FKOW, SIGMOD'23]

SJF + $\neg$
FO, non-**FO**
[KW, PODS'18]

Conjecture: For every (union of) BCQ q, CERTAINTY(q) is in **P** or **coNP**-complete.

Finding Consistent Answers from Inconsistent Data: Systems, Algorithms, and Complexity

kNN + missing values
[Karlaš et al., VLDB'21]

kNN + FD
P, **coNP**-complete
[FK, ICDT'22]

Bayes + missing
(BOFK, submitted)

ML + dirty data
...?

graph query
...**FO**, **P**$\setminus$ **FO**, **coNP**-complete?

SJF + (intgrty. const.)
...**FO**, non-**FO**?

SJF rooted trees (and beyond)
FO, **P**$\setminus$ **FO**, **coNP**-complete
[KOW, PODS'24]

SJF + PK & (unary)FK
FO, non-**FO**
[HW, PODS'22]

SJF path
FO, **NL**-complete, **P**-complete, **coNP**-complete
[KOW, PODS'21]

LinCQA$^+$
...SJF acyclic **FO** in $O(N)$?

SJF + multiple keys
FO, non-**FO**
[KW, PODS'20]

SJF
FO, **L**-complete, **coNP**-complete
[KW, ICDT'19]

LinCQA
PPJT via **FO** in $O(N)$
[FKOW, SIGMOD'23]

SJF + $\neg$
FO, non-**FO**
[KW, PODS'18]

Conjecture: For every (union of) BCQ q, CERTAINTY(q) is in **P** or **coNP**-complete.

Finding Consistent Answers from Inconsistent Data: Systems, Algorithms, and Complexity

kNN + missing values
[Karlaš et al., VLDB'21]

kNN + FD
P, **coNP**-complete
[FK, ICDT'22]

Bayes + missing
(BOFK, submitted)

ML + dirty data
. . . ?

graph query
. . . **FO**, **P**$\setminus$ **FO**, **coNP**-complete?

SJF + (intgrty. const.)
. . . **FO**, non-**FO**?

SJF rooted trees (and beyond)
FO, **P**$\setminus$ **FO**, **coNP**-complete
[KOW, PODS'24]

SJF + PK & (unary)FK
FO, non-**FO**
[HW, PODS'22]

SJF path
FO, **NL**-complete, **P**-complete, **coNP**-complete
[KOW, PODS'21]

LinCQA$^+$
. . . SJF acyclic **FO** in $O(N)$?

SJF + multiple keys
FO, non-**FO**
[KW, PODS'20]

SJF
FO, **L**-complete, **coNP**-complete
[KW, ICDT'19]

LinCQA
PPJT via **FO** in $O(N)$
[FKOW, SIGMOD'23]

SJF + $\neg$
FO, non-**FO**
[KW, PODS'18]

Conjecture: For every (union of) BCQ q, CERTAINTY(q) is in **P** or **coNP**-complete.

Finding Consistent Answers from Inconsistent Data: Systems, Algorithms, and Complexity

*k*NN + missing values
[Karlaš et al., VLDB'21]

*k*NN + FD
P, **coNP**-complete
[FK, ICDT'22]

Bayes + missing
(BOFK, submitted)

ML + dirty data
. . . ?

graph query
. . . **FO**, **P**$\setminus$ **FO**, **coNP**-complete?

SJF rooted trees (and beyond)
FO, **P**$\setminus$ **FO**, **coNP**-complete
[KOW, PODS'24]

SJF path
FO, **NL**-complete, **P**-complete, **coNP**-complete
[KOW, PODS'21]

SJF
FO, **L**-complete, **coNP**-complete
[KW, ICDT'19]

LinCQA$^+$
. . . SJF acyclic **FO** in $O(N)$?

LinCQA
PPJT via **FO** in $O(N)$
[FKOW, SIGMOD'23]

SJF + (intgrty. const.)
. . . **FO**, non-**FO**?

SJF + PK & (unary)FK
FO, non-**FO**
[HW, PODS'22]

SJF + multiple keys
FO, non-**FO**
[KW, PODS'20]

SJF + $\neg$
FO, non-**FO**
[KW, PODS'18]

Conjecture: For every (union of) BCQ q, CERTAINTY(q) is in **P** or **coNP**-complete.

Finding Consistent Answers from Inconsistent Data: Systems, Algorithms, and Complexity

kNN + missing values
[Karlaš et al., VLDB'21]

kNN + FD
P, **coNP**-complete
[FK, ICDT'22]

Bayes + missing
(BOFK, submitted)

ML + dirty data
...?

graph query
...**FO**, **P**$\setminus$ **FO**, **coNP**-complete?

SJF + (intgrty. const.)
...**FO**, non-**FO**?

SJF rooted trees (and beyond)
FO, **P**$\setminus$ **FO**, **coNP**-complete
[KOW, PODS'24]

SJF + PK & (unary)FK
FO, non-**FO**
[HW, PODS'22]

SJF path
FO, **NL**-complete, **P**-complete, **coNP**-complete
[KOW, PODS'21]

LinCQA$^+$
...SJF acyclic **FO** in $O(N)$?

SJF + multiple keys
FO, non-**FO**
[KW, PODS'20]

SJF
FO, **L**-complete, **coNP**-complete
[KW, ICDT'19]

LinCQA
PPJT via **FO** in $O(N)$
[FKOW, SIGMOD'23]

SJF + $\neg$
FO, non-**FO**
[KW, PODS'18]

Conjecture: For every (union of) BCQ q, CERTAINTY(q) is in **P** or **coNP**-complete.

Finding Consistent Answers from Inconsistent Data: Systems, Algorithms, and Complexity

kNN + missing values
[Karlaš et al., VLDB'21]

kNN + FD
P, **coNP**-complete
[FK, ICDT'22]

Bayes + missing
(BOFK, submitted)

ML + dirty data
...?

graph query
...**FO**, **P**$\setminus$ **FO**, **coNP**-complete?

SJF rooted trees (and beyond)
FO, **P**$\setminus$ **FO**, **coNP**-complete
[KOW, PODS'24]

SJF path
FO, **NL**-complete, **P**-complete, **coNP**-complete
[KOW, PODS'21]

SJF
FO, **L**-complete, **coNP**-complete
[KW, ICDT'19]

LinCQA^{+}
...SJF acyclic **FO** in $O(N)$?

LinCQA
PPJT via **FO** in $O(N)$
[FKOW, SIGMOD'23]

SJF + (intgrty. const.)
...**FO**, non-**FO**?

SJF + PK & (unary)FK
FO, non-**FO**
[HW, PODS'22]

SJF + multiple keys
FO, non-**FO**
[KW, PODS'20]

SJF + $\neg$
FO, non-**FO**
[KW, PODS'18]

Conjecture: For every (union of) BCQ q, CERTAINTY(q) is in **P** or **coNP**-complete.

Finding Consistent Answers from Inconsistent Data: Systems, Algorithms, and Complexity

kNN + missing values
[Karlaš et al., VLDB'21]

kNN + FD
P, **coNP**-complete
[FK, ICDT'22]

Bayes + missing
(BOFK, submitted)

ML + dirty data
. . . ?

graph query
. . . **FO**, **P**$\setminus$ **FO**, **coNP**-complete?

SJF + (intgrty. const.)
. . . **FO**, non-**FO**?

SJF rooted trees (and beyond)
FO, **P**$\setminus$ **FO**, **coNP**-complete
[KOW, PODS'24]

SJF + PK & (unary)FK
FO, non-**FO**
[HW, PODS'22]

SJF path
FO, **NL**-complete, **P**-complete, **coNP**-complete
[KOW, PODS'21]

LinCQA$^+$
. . . SJF acyclic **FO** in $O(N)$?

SJF + multiple keys
FO, non-**FO**
[KW, PODS'20]

SJF
FO, **L**-complete, **coNP**-complete
[KW, ICDT'19]

LinCQA
PPJT via **FO** in $O(N)$
[FKOW, SIGMOD'23]

SJF + $\neg$
FO, non-**FO**
[KW, PODS'18]

Conjecture: For every (union of) BCQ q, CERTAINTY(q) is in **P** or **coNP**-complete.

Finding Consistent Answers from Inconsistent Data: Systems, Algorithms, and Complexity

kNN + missing values
[Karlaš et al., VLDB'21]

kNN + FD
P, **coNP**-complete
[FK, ICDT'22]

Bayes + missing
(BOFK, submitted)

ML + dirty data
. . . ?

graph query
. . . **FO**, **P**\ **FO**, **coNP**-complete?

SJF + (intgrty. const.)
. . . **FO**, non-**FO**?

SJF rooted trees (and beyond)
FO, **P**\ **FO**, **coNP**-complete
[KOW, PODS'24]

SJF + PK & (unary)FK
FO, non-**FO**
[HW, PODS'22]

SJF path
FO, **NL**-complete, **P**-complete, **coNP**-complete
[KOW, PODS'21]

LinCQA^{+}
. . . SJF acyclic **FO** in $O(N)$?

SJF + multiple keys
FO, non-**FO**
[KW, PODS'20]

SJF
FO, **L**-complete, **coNP**-complete
[KW, ICDT'19]

LinCQA
PPJT via **FO** in $O(N)$
[FKOW, SIGMOD'23]

SJF + $\neg$
FO, non-**FO**
[KW, PODS'18]

Conjecture: For every (union of) BCQ q, CERTAINTY(q) is in **P** or **coNP**-complete.

Finding Consistent Answers from Inconsistent Data: Systems, Algorithms, and Complexity

kNN + missing values
[Karlaš et al., VLDB'21]

kNN + FD
P, **coNP**-complete
[FK, ICDT'22]

Bayes + missing
(BOFK, submitted)

ML + dirty data
...?

graph query
...**FO**, **P**$\setminus$ **FO**, **coNP**-complete?

SJF + (intgrty. const.)
...**FO**, non-**FO**?

SJF rooted trees (and beyond)
FO, **P**$\setminus$ **FO**, **coNP**-complete
[KOW, PODS'24]

SJF + PK & (unary)FK
FO, non-**FO**
[HW, PODS'22]

SJF path
FO, **NL**-complete, **P**-complete, **coNP**-complete
[KOW, PODS'21]

LinCQA$^+$
...SJF acyclic **FO** in $O(N)$?

SJF + multiple keys
FO, non-**FO**
[KW, PODS'20]

SJF
FO, **L**-complete, **coNP**-complete
[KW, ICDT'19]

LinCQA
PPJT via **FO** in $O(N)$
[FKOW, SIGMOD'23]

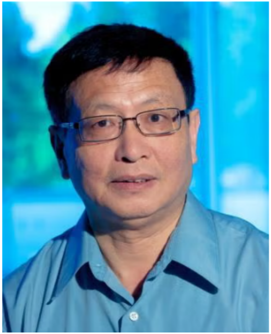
SJF + $\neg$
FO, non-**FO**
[KW, PODS'18]

Conjecture: For every (union of) BCQ q, CERTAINTY(q) is in **P** or **coNP**-complete.

The Beauty of Bounded Gaps

A huge discovery about prime numbers—and what it means for the future of math.

BY JORDAN ELLENBERG MAY 22, 2013 • 4:44 PM



Thank YOU!

Uri Andrews, Jin-Yi Cai, Paris Koutris, Jignesh Patel, Jef Wijsen

Yixin Cao, Rocky K. C. Chang, AnHai Doan, Steve Foote, Wei-Chiao Hsu, Alekh Jindal, Phokion Kolaitis, Ren Mao, Jeff Naughton, Hung Ngo, Lowell Rausch, Abhishek Roy, Ning Tan, Angela Thorp, Kristen Tinetti, Bin Xu, Fan (Amy) Yang

Song Bian, Ting Cai, Bing An Chang, Xufeng Cai, Elvis Chang, Jiang Chang, Kaiyang Chen, Maggie Chen, Yiding Chen, Nick Corrado, Shaleen Deep, Austen Z. Fan, Yuhang Fan, Zhiwei Fan, Kevin Gaffney, Yue Gao, Evangelia Gergatsouli, Jinshan Gu, Xinyu Guan, Yang Guo, Ankur Goswami, Yilin He, Hengjing Huang, Shunyi Huang, Aarati Kakaraparthy, Yuping Ke, Fengan Li, Justin LiXie, Holdson Liang, Eric Lin, Derek Ma, Jeremy McMahan, Simiao Ren, Yue Shi, Kartik Sreenivasan, Xiaoxi Sun, Yuxin Sun, Remy Wang, Xiang Wang, Jingcheng Xu, Jie You, Peng Yu, Zhe Zeng, Jifan Zhang, Ling Zhang, Hangdong Zhao, Xingjian Zhen, Yi Zhou

Yufei Gao, Feng-Ying Ma, Hong Ouyang, Zhong-Zhan Ouyang, Yujia Peng, Weisheng Wang, Hao Wu, Bin Xia, Yifan Xia

Finding Consistent Answers from Inconsistent Data: Systems, Algorithms, and Complexity

kNN + missing values
[Karlaš et al., VLDB'21]

kNN + FD
P, **coNP**-complete
[FK, ICDT'22]

Bayes + missing
(BOFK, submitted)

ML + dirty data
...?

graph query
...**FO**, **P**\ **FO**, **coNP**-complete?

SJF + (intgrty. const.)
...**FO**, non-**FO**?

SJF rooted trees (and beyond)
FO, **P**\ **FO**, **coNP**-complete
[KOW, PODS'24]

SJF + PK & (unary)FK
FO, non-**FO**
[HW, PODS'22]

SJF path
FO, **NL**-complete, **P**-complete, **coNP**-complete
[KOW, PODS'21]

LinCQA$^+$
...SJF acyclic **FO** in $O(N)$?

SJF + multiple keys
FO, non-**FO**
[KW, PODS'20]

SJF
FO, **L**-complete, **coNP**-complete
[KW, ICDT'19]

LinCQA
PPJT via **FO** in $O(N)$
[FKOW, SIGMOD'23]

SJF + $\neg$
FO, non-**FO**
[KW, PODS'18]

Conjecture: For every (union of) BCQ q, CERTAINTY(q) is in **P** or **coNP**-complete.