Consistent Query Answering for Primary Keys on Path Queries

Paris Koutris [1] Xiating Ouyang [1] Jef Wijsen [2]

University of Wisconsin–Madison [1]

University of Mons [2]

PODS, Xi'an Shaanxi, China, June 20–25 2021

Data model

- Primary key constraint as the only integrity constraint
- Inconsistent relational databases violating the primary constraint
- A repair is an inclusion-maximal consistent subinstance

Univ

Acronym	City
UW	Madison
UW	Seattle
UMONS	Mons

Weather

City	Weather
Madison	Snow
Seattle	Rain
Mons	Sunny

Q: Is there a university snowing today?

$$Q() : -\text{Univ}(\underline{x}, y), \text{Weather}(\underline{y}, \texttt{`Snow'})$$

The problem statement

Consistent query answering – CERTAINTY(q)

INPUT: an inconsistent database db
QUESTION: Is the **fixed** Boolean query q true in <u>all</u> repairs of db?

$q_2() : -R(\underline{x}, z), S(\underline{y}, z)$

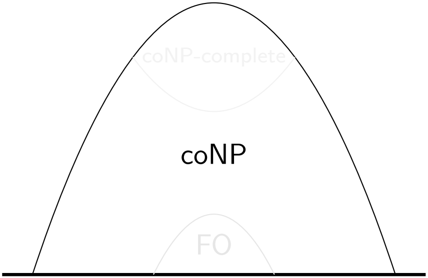
$q_1() : -R(\underline{x}, y), S(\underline{y}, z)$

$$\text{CERTAINTY}(q_1) \iff \exists x(\exists y R(\underline{x}, y) \land \forall y R(\underline{x}, y) \rightarrow \exists z S(\underline{y}, z))$$

The problem statement

Consistent query answering – CERTAINTY(q)

INPUT:	an inconsistent database db
QUESTION:	Is the **fixed** Boolean query q true in all repairs of db?



$q_2() : -R(\underline{x}, z), S(\underline{y}, z)$

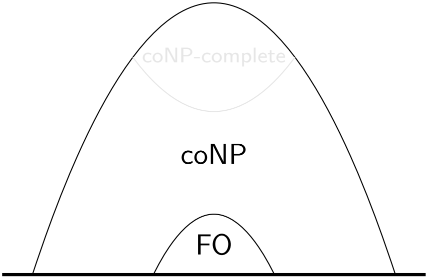
$q_1() : -R(\underline{x}, y), S(\underline{y}, z)$

$$\text{CERTAINTY}(q_1) \Longleftrightarrow \exists x(\exists y R(\underline{x}, y) \land \forall y R(\underline{x}, y) \rightarrow \exists z S(\underline{y}, z))$$

The problem statement

Consistent query answering – CERTAINTY(q)

INPUT:	an inconsistent database db
QUESTION:	Is the **fixed** Boolean query q true in <u>all</u> repairs of db?



$$q_2() : -R(\underline{x}, z), S(\underline{y}, z)$$

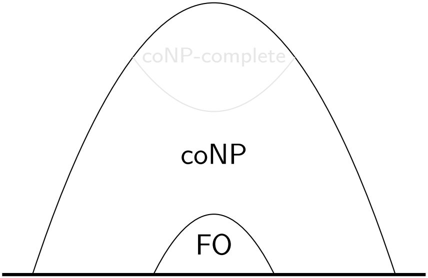
$$q_1() : -R(\underline{x}, y), S(\underline{y}, z)$$

$$\text{CERTAINTY}(q_1) \iff \exists x(\exists y R(\underline{x}, y) \wedge \forall y R(\underline{x}, y) \rightarrow \exists z S(\underline{y}, z))$$

The problem statement

Consistent query answering – CERTAINTY(q)

INPUT:	an inconsistent database db
QUESTION:	Is the **fixed** Boolean query q true in <u>all</u> repairs of db?



$q_2() :- R(\underline{x}, z), S(\underline{y}, z)$

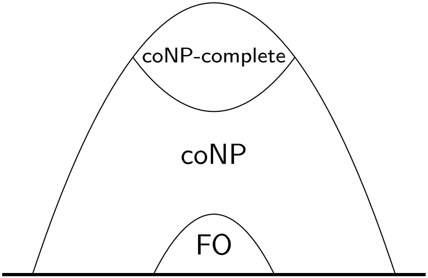
$q_1() :- R(\underline{x}, y), S(\underline{y}, z)$

$$\text{CERTAINTY}(q_1) \Longleftrightarrow \exists x(\exists y R(\underline{x}, y) \wedge \forall y R(\underline{x}, y) \rightarrow \exists z S(\underline{y}, z))$$

The problem statement

Consistent query answering – CERTAINTY(q)

INPUT:	an inconsistent database db
QUESTION:	Is the **fixed** Boolean query q true in <u>all</u> repairs of db?



$$q_2() :- R(\underline{x}, z), S(\underline{y}, z)$$

$$q_1() :- R(\underline{x}, y), S(\underline{y}, z)$$

$$\text{CERTAINTY}(q_1) \iff \exists x(\exists y R(\underline{x}, y) \wedge \forall y R(\underline{x}, y) \rightarrow \exists z S(\underline{y}, z))$$

Complexity classification

Conjecture

For any BCQ q, $\mathsf{CERTAINTY}(q)$ *is either in* PTIME *or* coNP*-complete.*

Complexity classification

Conjecture

For any BCQ q, CERTAINTY(q) *is either in* PTIME *or* coNP*-complete.*

Classification	BCQ Class	Result
FO, non-**FO**	$\mathcal{C}_{forest}$ (SJF$_{\text{(self-join-free)}}$)	[Fuxman and Miller, ICDT'05]
FO, non-**FO**	SJF α-acyclic queries	[Wijsen, PODS'10]
P, **coNP**-comp.	SJF two atoms	[Kolaitis and Pema, 12]
P, **coNP**-comp.	SJF simple keys	[Koutris and Suciu, ICDT'14]
FO, **P** $\setminus$ **FO**, **coNP**-comp.	SJF	[Koutris and Wijsen, PODS'15]
FO, **L**-comp., **coNP**-comp. $\star$	SJF	[Koutris and Wijsen, ICDT'19]
FO	SJF path	(implied by [KW, PODS'15])

Complexity classification

Conjecture

For any BCQ q, CERTAINTY(q) *is either in* PTIME *or* coNP*-complete.*

Classification	BCQ Class	Result
FO, non-**FO**	$\mathcal{C}_{forest}$ (SJF$_{\text{(self-join-free)}}$)	[Fuxman and Miller, ICDT'05]
FO, non-**FO**	SJF α-acyclic queries	[Wijsen, PODS'10]
P, **coNP**-comp.	SJF two atoms	[Kolaitis and Pema, 12]
P, **coNP**-comp.	SJF simple keys	[Koutris and Suciu, ICDT'14]
FO, **P** $\setminus$ **FO**, **coNP**-comp.	SJF	[Koutris and Wijsen, PODS'15]
FO, **L**-comp., **coNP**-comp. $\star$	SJF	[Koutris and Wijsen, ICDT'19]
FO $\leftarrow$	SJF path	(implied by [KW, PODS'15])

$$q() : -R_1(\underline{x_1}, x_2), R_2(\underline{x_2}, x_3), \cdots, R_n(\underline{x_n}, x_{n+1}) \quad \Longrightarrow \quad R_1 R_2 \cdots R_n$$

distinct variables x_i, relation names R_i

Complexity classification

Conjecture

For any BCQ q, CERTAINTY(q) *is either in* PTIME *or* coNP*-complete.*

Classification	BCQ Class	Result
FO, non-**FO**	$\mathcal{C}_{forest}$ (SJF$_{\text{(self-join-free)}}$)	[Fuxman and Miller, ICDT'05]
FO, non-**FO**	SJF α-acyclic queries	[Wijsen, PODS'10]
P, **coNP**-comp.	SJF two atoms	[Kolaitis and Pema, 12]
P, **coNP**-comp.	SJF simple keys	[Koutris and Suciu, ICDT'14]
FO, **P** $\setminus$ **FO**, **coNP**-comp.	SJF	[Koutris and Wijsen, PODS'15]
FO, **L**-comp., **coNP**-comp. $\star$	SJF	[Koutris and Wijsen, ICDT'19]
FO $\leftarrow$	SJF path	(implied by [KW, PODS'15])
FO, **NL**-comp., **P**-comp., **coNP**-comp.	SJF **path**	**(Our result)**

$$q() :- R_1(\underline{x_1}, x_2), R_2(\underline{x_2}, x_3), \cdots, R_n(\underline{x_n}, x_{n+1}) \quad \Longrightarrow \quad R_1 R_2 \cdots R_n$$

distinct variables x_i, relation names R_i

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

The curse of self-joins

Theorem (Deletion Propagation, TODS 2012)

For every CQ without self-joins, deletion propagation is either APX-hard or solvable (in polynomial time) by the unidimensional algorithm.

Theorem (Pricing, JACM 2015)

Let Q be a CQ without self-joins. The data complexity for PRICE(Q) *is either in* PTIME *or* NP*-complete.*

Theorem (Query resilience, PODS 2020)

Let q be a single-self-join-CQ with at most two occurrences of the self-join relation. The problem RES(q) *is either in* PTIME *or* NP*-complete.*

Theorem (Our result)

Let q be a path query. The problem CERTAINTY(q) *is either in* FO, NL*-complete,* PTIME*-complete or* coNP*-complete.*

The curse of self-joins

Theorem (Deletion Propagation, TODS 2012)

For every CQ without self-joins, deletion propagation is either APX-hard or solvable (in polynomial time) by the unidimensional algorithm.

Theorem (Pricing, JACM 2015)

Let Q be a CQ without self-joins. The data complexity for PRICE(Q) *is either in* PTIME *or* NP*-complete.*

Theorem (Query resilience, PODS 2020)

Let q be a single-self-join-CQ with at most two occurrences of the self-join relation. The problem RES(q) *is either in* PTIME *or* NP*-complete.*

Theorem (Our result)

Let q be a path query. The problem CERTAINTY(q) *is either in* FO, NL*-complete,* PTIME*-complete or* coNP*-complete.*

Outline

1. Handling self-joins
2. Classification result
3. Proof sketch

Outline

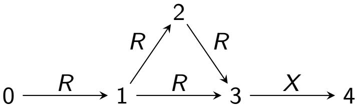
1. Handling self-joins
2. Classification result
3. Proof sketch

The notion of "rewinding"

INPUT to CERTAINTY(RRX):

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

X	$\underline{B_1}$	B_2
	3	4



QUESTION: Do all repairs contain an RRX path?

r_1

RRX *RRX* *RRX*

r_2

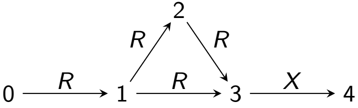
RRX *RRX*

RRX *RRX* *RRX*

INPUT to CERTAINTY(RRX):

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

X	$\underline{B_1}$	B_2
	3	4



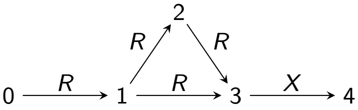
QUESTION: Do all repairs contain an RRX path?

The notion of "rewinding"

INPUT to CERTAINTY(RRX):

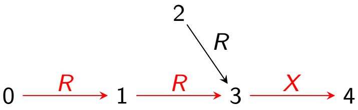
R	$\underline{A_1}$	A_2
	0	1
	1	2
	1	3
	2	3

X	$\underline{B_1}$	B_2
	3	4



QUESTION: Do all repairs contain an RRX path?

r_1



RRX RRX RRX

r_2

RRX RRX

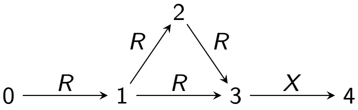
RRX RRX RRX

The notion of "rewinding"

INPUT to CERTAINTY(RRX):

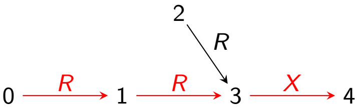
R	$\underline{A_1}$	A_2
	0	1
	1	2
	1	3
	2	3

X	$\underline{B_1}$	B_2
	3	4



QUESTION: Do all repairs contain an RRX path?

r$_1$



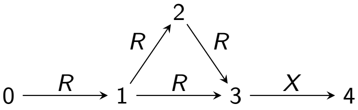
RRX RRX RRX

r$_2$

INPUT to CERTAINTY(RRX):

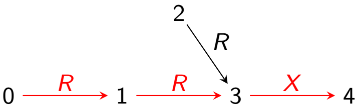
R	$\underline{A_1}$	A_2
	0	1
	1	2
	1	3
	2	3

X	$\underline{B_1}$	B_2
	3	4



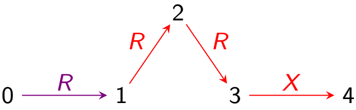
QUESTION: Do all repairs contain an RRX path?

r_1



RRX RRX RRX

r_2



RRX RRX

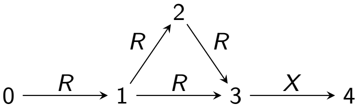
RRX RRX RRX

The notion of "rewinding"

INPUT to CERTAINTY(RRX):

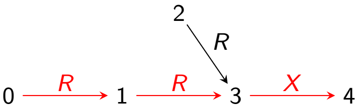
R	$\underline{A_1}$	A_2
	0	1
	1	2
	1	3
	2	3

X	$\underline{B_1}$	B_2
	3	4



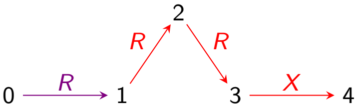
QUESTION: Do all repairs contain an RRX path?

r_1



RRX $\quad$ RRX $\quad$ RRX

r_2



RRX $\quad$ RRX

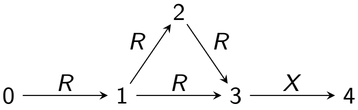
RRX $\quad$ RRX $\quad$ RRX

The notion of "rewinding"

INPUT to CERTAINTY(RRX):

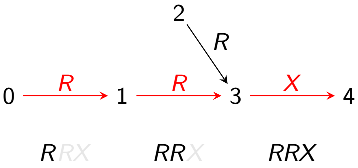
R	$\underline{A_1}$	A_2
	0	1
	1	2
	1	3
	2	3

X	$\underline{B_1}$	B_2
	3	4

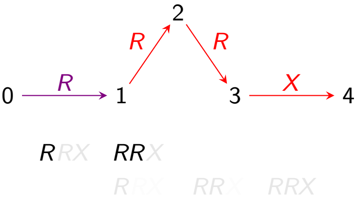


QUESTION: Do all repairs contain an RRX path?

r_1



r_2

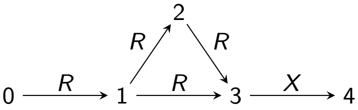


The notion of "rewinding"

INPUT to CERTAINTY(RRX):

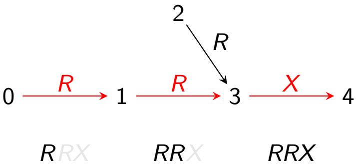
R	$\underline{A_1}$	A_2
	0	1
	1	2
	1	3
	2	3

X	$\underline{B_1}$	B_2
	3	4

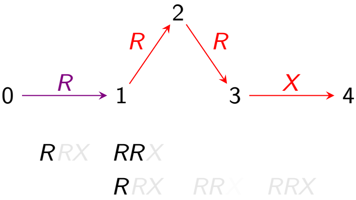


QUESTION: Do all repairs contain an RRX path?

r$_1$



r$_2$

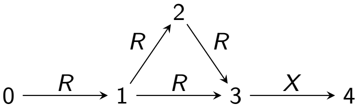


The notion of “rewinding”

INPUT to CERTAINTY(RRX):

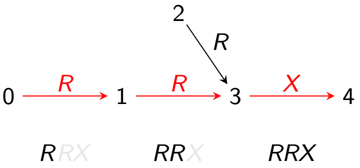
R	$\underline{A_1}$	A_2
	0	1
	1	2
	1	3
	2	3

X	$\underline{B_1}$	B_2
	3	4

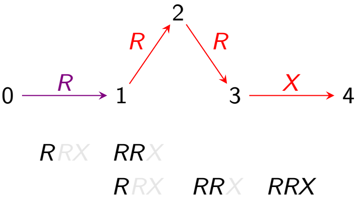


QUESTION: Do all repairs contain an RRX path?

r$_1$



r$_2$



The notion of “rewinding” (cont.)

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.

“Reachability”, “**NL**-complete”

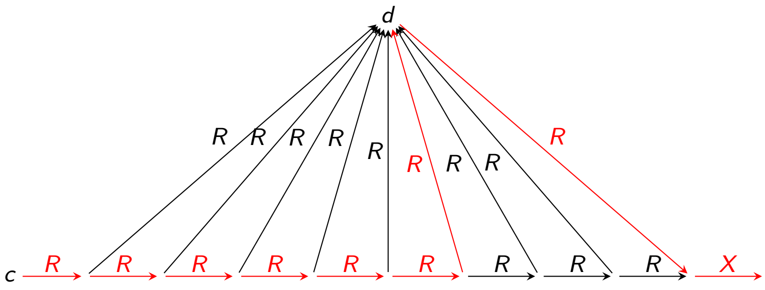
How to find the regular expression?

The notion of "rewinding" (cont.)

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.



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

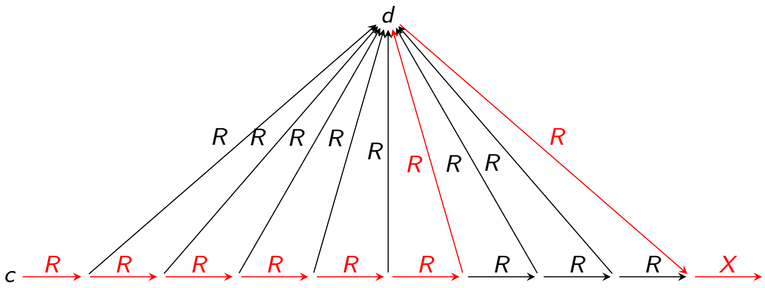
How to find the regular expression?

The notion of “rewinding” (cont.)

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.



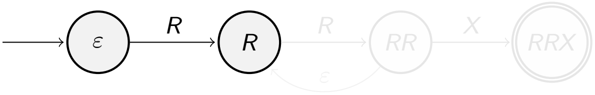
“Reachability”, “**NL**-complete”

How to find the regular expression?

From path query to NFA

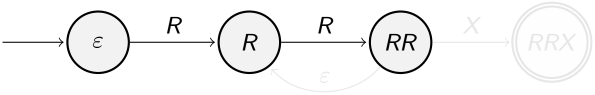
NFA(RRX)

From path query to NFA



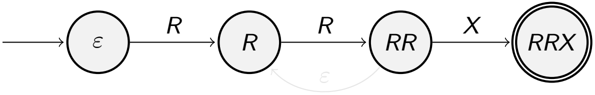
NFA(RRX)

From path query to NFA



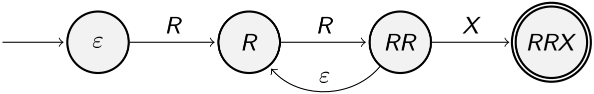
NFA(RRX)

From path query to NFA



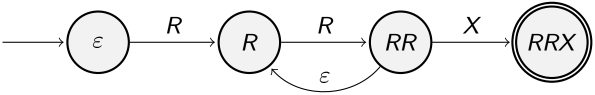
NFA(RRX)

From path query to NFA



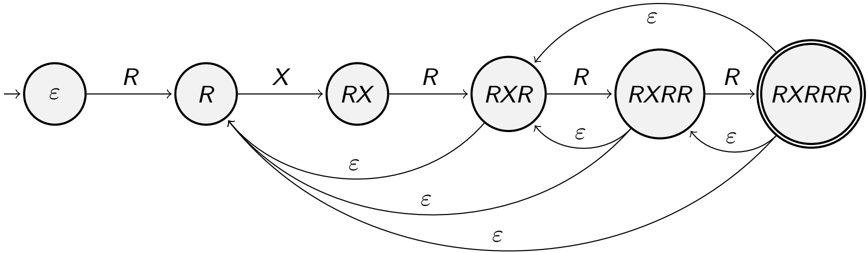
NFA(RRX)

From path query to NFA



NFA(RRX)

From path query to NFA (cont.)



NFA(*RXRRR*)

Outline

Our result

NL-hard

$q_2 = RXRY$ $\qquad$ $RXRXRY \in \mathrm{NFA}(q_2)$

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

$q_1 = RXRX$ $\qquad$ $RXRXRX \in \mathrm{NFA}(q_1)$

FO-rewritable

Our result

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

C_3: q is a factor of every word in $\text{NFA}(q)$

PTIME

NL-hard

$q_2 = RXRY$ $RXRXRY \in \text{NFA}(q_2)$

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

$q_1 = RXRX$ $RXRXRX \in \text{NFA}(q_1)$

FO-rewritable

Our result

coNP-complete

$q_4 = RXRXRYRY \qquad RXRXRYRXRYRY \in \mathsf{NFA}(q_4)$

C_3: q is a factor of every word in $\mathsf{NFA}(q)$

PTIME

$q_3 = RXRYRY$

C_2: 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

$q_2 = RXRY \qquad RXRXRY \in \mathsf{NFA}(q_2)$

C_1: q is a prefix of every word in $\mathsf{NFA}(q)$

$q_1 = RXRX \qquad RXRXRX \in \mathsf{NFA}(q_1)$

FO-rewritable

Our result

coNP-complete

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

C$_3$: q is a factor of every word in NFA(q)

PTIME-complete

$q_3 = RXRYRY$

C$_2$: 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

$q_2 = RXRY \qquad RXRXRY \in \text{NFA}(q_2)$

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

$q_1 = RXRX \qquad RXRXRX \in \text{NFA}(q_1)$

FO-rewritable

C_1, C_2 and C_3 are decidable

C_1 : q is a prefix of every word in NFA(q)

$\Longleftrightarrow$ Whenever $q = u \cdot Rv \cdot Rw$, q is a prefix of $u \cdot Rv \cdot Rv \cdot Rw$.

C_3 : q is a factor of every word in NFA(q)

$\Longleftrightarrow$ Whenever $q = u \cdot Rv \cdot Rw$, q is a factor of $u \cdot Rv \cdot Rv \cdot Rw$.

Outline

Lemma (**PTIME**)

Let q be a path query satisfying C_3. 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 at c.

Moreover, item 2 can be decided in **PTIME** using dynamic programming/least fixedpoint logic.

C_3: q is a factor of every word in NFA(q)

Lemma (PTIME)

Let q be a path query satisfying C_3. 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 at c.

Moreover, item 2 can be decided in **PTIME** using dynamic programming/least fixedpoint logic.

C_3: q is a factor of every word in NFA(q)

Lemma (**PTIME**)

Let q be a path query satisfying C_3. 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 at c.

Moreover, item 2 can be decided in **PTIME** using dynamic programming/least fixedpoint logic.

C_3: q is a factor of every word in NFA(q)

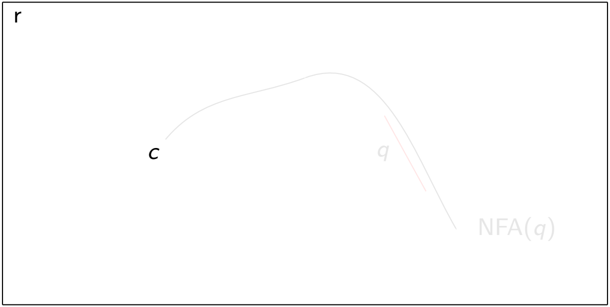
Lemma (**PTIME**)

Let q be a path query satisfying C_3. 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 at c.

Moreover, item 2 can be decided in **PTIME** using dynamic programming/least fixedpoint logic.

C_3: q is a factor of every word in NFA(q)



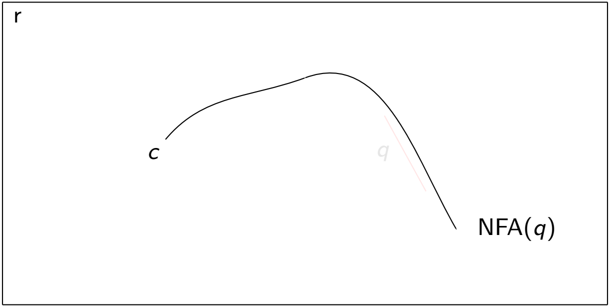
Lemma (**PTIME**)

Let q be a path query satisfying C_3. 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 at c.

Moreover, item 2 can be decided in **PTIME** using dynamic programming/least fixedpoint logic.

C_3: q is a factor of every word in NFA(q)



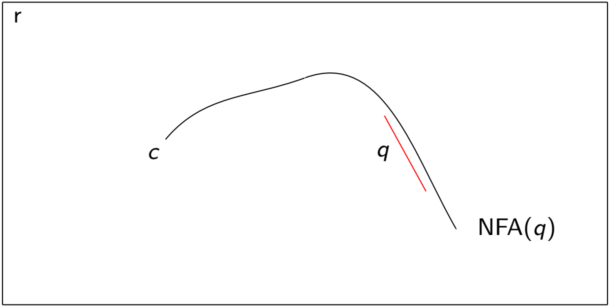
Lemma (**PTIME**)

Let q be a path query satisfying C_3. 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 at c.

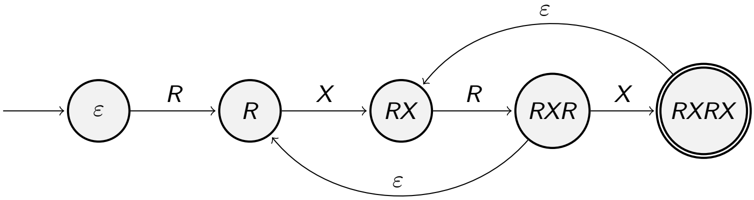
Moreover, item 2 can be decided in **PTIME** using dynamic programming/least fixedpoint logic.

C_3: q is a factor of every word in NFA(q)



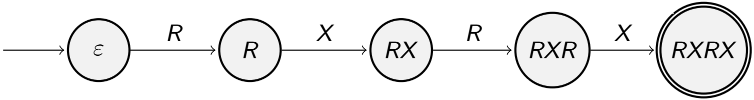
FO-rewritability

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



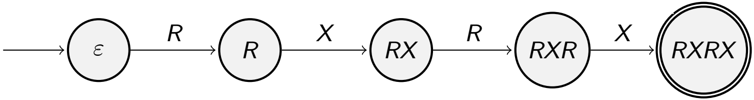
FO-rewritability

C_1: *q is a prefix of every word in* NFA(q)



FO-rewritability

C_1: *q is a prefix of every word in* NFA(q)



Lemma (**FO**)

Let q be a path query satisfying C_1. 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 of $\underline{q}$ starting at c.

Moreover, item 2 can be decided in **FO**.

Hardness results

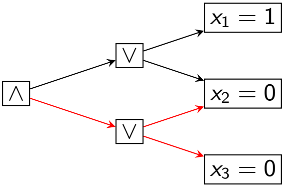
Lemma

Let q be a path query. Then we have

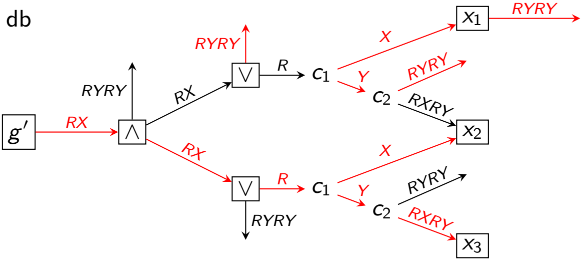
- if q does not satisfy C_1, then CERTAINTY(q) is **NL**-hard (via Reachability);
- if q does not satisfy C_2, then CERTAINTY(q) is **PTIME**-hard (via Monotone Circuit Value); and
- if q does not satisfy C_3, then CERTAINTY(q) is **coNP**-hard (via Satisfiability).

$$q = \underbrace{\quad}_{u}\underbrace{RX}_{Rv_1}\underbrace{RY}_{Rv_2}\underbrace{RY}_{Rw}$$

$\mathcal{C}$



db



Future works

Conjecture

Let q be a CQ. Then we have

- CERTAINTY(q) *is either in* PTIME *or* coNP*-complete, and it is decidable which of the two cases applies; and*
- *it is decidable whether or not* CERTAINTY(q) *is in* FO*.*

- Acyclic queries with self-joins
- Multiple key constraints, negated atom, aggregation etc.

Conclusion

coNP-complete	$q_4 = RXRXRYRY$	$RXRXRYRXRYRY \in \mathrm{NFA}(q_4)$

C$_3$: q is a factor of every word in NFA(q)

PTIME-complete	$q_3 = RXRYRY$	

C$_2$: 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	$q_2 = RXRY$	$RXRXRY \in \mathrm{NFA}(q_2)$

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

FO-rewritable	$q_1 = RXRX$	$RXRXRX \in \mathrm{NFA}(q_1)$