Datalog as a Query Language for Data Exchange Systems

Marcelo Arenas

PUC Chile

Joint work with Pablo Barceló and Juan Reutter

Outline

- The data exchange scenario
  - Query answering
- Queries with negation in data exchange
- $\text{DATALOG}^{\mathbf{C}(\neq)}$ programs
- Beyond union of conjunctive queries
  - Expressive power of $\text{DATALOG}^{\mathbf{C}(\neq)}$
  - New tractable classes of queries
- Concluding remarks

Outline

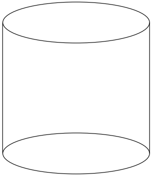
- The data exchange scenario
  - Query answering
- Queries with negation in data exchange
- $\textsc{Datalog}^{\mathbf{C}(\neq)}$ programs
- Beyond union of conjunctive queries
  - Expressive power of $\textsc{Datalog}^{\mathbf{C}(\neq)}$
  - New tractable classes of queries
- Concluding remarks

The data exchange scenario

$$\text{Source Schema } \mathbf{S} \xrightarrow{\quad \boldsymbol{\Sigma} \quad} \text{Target Schema } \mathbf{T}$$

The data exchange scenario

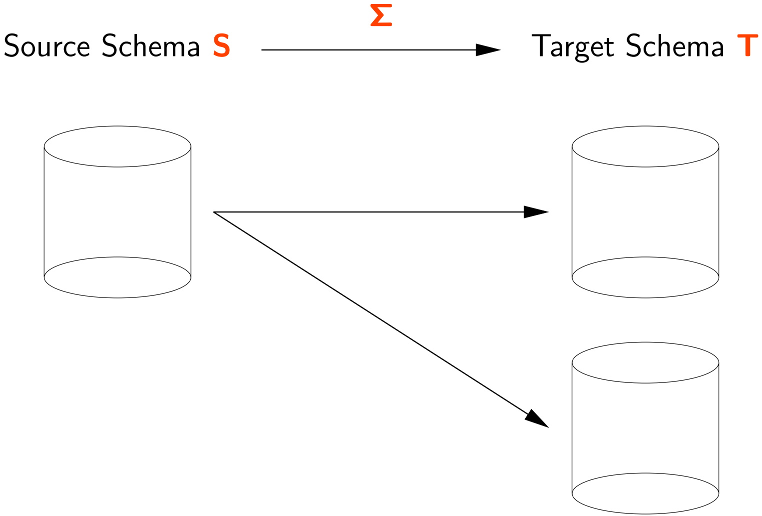
Source Schema **S** $\xrightarrow{\boldsymbol{\Sigma}}$ Target Schema **T**



The data exchange scenario

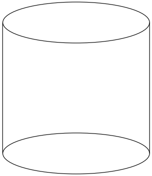
Source Schema **S** $\xrightarrow{\mathbf{\Sigma}}$ Target Schema **T**

The data exchange scenario



The data exchange scenario

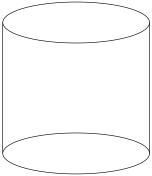
Source Schema **S** $\xrightarrow{\boldsymbol{\Sigma}}$ Target Schema **T**



Query **Q**

The data exchange scenario

Source Schema **S** $\xrightarrow{\boldsymbol{\Sigma}}$ Target Schema **T**

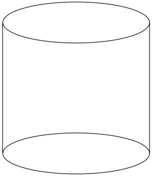


Query **Q**

What is the semantics of **Q**?

The data exchange scenario

Source Schema $\mathbf{S}$ $\xrightarrow{\Sigma}$ Target Schema $\mathbf{T}$



Query $\mathbf{Q}$

What is the semantics of $\mathbf{Q}$?

Can $\mathbf{Q}$ be evaluated efficiently?

Data exchange settings

Data exchange setting: $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$

- $\mathbf{S}$: source schema
- $\mathbf{T}$: target schema
- Σ: set of source-to-target dependencies

Data exchange settings

Data exchange setting: $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$

- $\mathbf{S}$: source schema
- $\mathbf{T}$: target schema
- Σ: set of source-to-target dependencies

Source-to-target dependency:

$$\forall \bar{x} \forall \bar{y} \left(\varphi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\, \psi(\bar{x}, \bar{z}) \right)$$

$\varphi(\bar{x}, \bar{y})$: conjunction of relational atomic formulas over $\mathbf{S}$
$\psi(\bar{x}, \bar{z})$: conjunction of relational atomic formulas over $\mathbf{T}$

Example: Data exchange setting

S: $Book(Title, AuthorName, Affiliation)$

T: $Writer(Name, BookTitle, Year)$

Σ:

$$Book(x_1, x_2, y_1) \quad \rightarrow \quad \exists z_1\, Writer(x_2, x_1, z_1)$$

Data exchange problem

Given a source instance I, find a target instance J such that (I, J) satisfies Σ.

- (I, J) satisfies $\varphi(\bar{x}, \bar{y}) \to \exists \bar{z}\, \psi(\bar{x}, \bar{z})$ if for every $(\bar{a}, \bar{b})$ such that I satisfies $\varphi(\bar{a}, \bar{b})$, there is a tuple $\bar{c}$ such that J satisfies $\psi(\bar{a}, \bar{c})$.

Data exchange problem

Given a source instance I, find a target instance J such that (I, J) satisfies Σ.

- (I, J) satisfies $\varphi(\bar{x}, \bar{y}) \to \exists \bar{z}\, \psi(\bar{x}, \bar{z})$ if for every $(\bar{a}, \bar{b})$ such that I satisfies $\varphi(\bar{a}, \bar{b})$, there is a tuple $\bar{c}$ such that J satisfies $\psi(\bar{a}, \bar{c})$.

J is a solution for I

Example: Data exchange problem

Previous example: $Book(x_1, x_2, y_1) \rightarrow \exists z_1\, Writer(x_2, x_1, z_1)$

I :

Book	*Title*	*AuthorName*	*Affiliation*
	Algebra	Hungerford	U. Washington
	Real Analysis	Royden	Stanford

Possible solutions:

Example: Data exchange problem

Previous example: $Book(x_1, x_2, y_1) \rightarrow \exists z_1\, Writer(x_2, x_1, z_1)$

I :

Book	*Title*	*AuthorName*	*Affiliation*
	Algebra	Hungerford	U. Washington
	Real Analysis	Royden	Stanford

Possible solutions:

J_1 :

Writer	*Name*	*BookTitle*	*Year*
	Hungerford	Algebra	1974
	Royden	Real Analysis	1988

Example: Data exchange problem

Previous example: $Book(x_1, x_2, y_1) \rightarrow \exists z_1\, Writer(x_2, x_1, z_1)$

I :

Book	*Title*	*AuthorName*	*Affiliation*
	Algebra	Hungerford	U. Washington
	Real Analysis	Royden	Stanford

Possible solutions:

J_1 :

Writer	*Name*	*BookTitle*	*Year*
	Hungerford	Algebra	1974
	Royden	Real Analysis	1988

J_2 :

Writer	*Name*	*BookTitle*	*Year*
	Hungerford	Algebra	$\perp_1$
	Royden	Real Analysis	$\perp_2$

Outline

- The data exchange scenario
  - Query answering
- Queries with negation in data exchange
- $\text{Datalog}^{\mathbf{C}(\neq)}$ programs
- Beyond union of conjunctive queries
  - Expressive power of $\text{Datalog}^{\mathbf{C}(\neq)}$
  - New tractable classes of queries
- Concluding remarks

Query answering in data exchange

Given: Data exchange setting $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, a query Q over $\mathbf{T}$ and an instance I of $\mathbf{S}$.

- What does it mean to answer Q?

Query answering in data exchange

Given: Data exchange setting $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, a query Q over $\mathbf{T}$ and an instance I of $\mathbf{S}$.

- What does it mean to answer Q?

$$\textsc{Certain}_{\mathcal{M}}(Q, I) \; = \bigcap_{J \text{ is a solution for } I} Q(J)$$

Example: Query answering in data exchange

Previous example: $Book(x_1, x_2, y_1) \rightarrow \exists z_1\, Writer(x_2, x_1, z_1)$

I :

Book	*Title*	*AuthorName*	*Affiliation*
	Algebra	Hungerford	U. Washington
	Real Analysis	Royden	Stanford

$$\textsc{Certain}_{\mathcal{M}}(\exists y \exists z\, Writer(x, y, z),\ I) \;=\; \{\text{Hungerford},\ \text{Royden}\}$$

Computing certain answers

- A data exchange setting $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ and a query Q are assumed to be fixed.
- Problem to solve:

Input	:	Instance I of $\mathbf{S}$ and a tuple $\bar{t}$ from I
Question	:	Is $\bar{t} \in \textsc{Certain}_{\mathcal{M}}(Q, I)$?

Computing certain answers

- A data exchange setting $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ and a query Q are assumed to be fixed.
- Problem to solve:

Input	:	Instance I of $\mathbf{S}$ and a tuple $\bar{t}$ from I
Question	:	Is $\bar{t} \in \textsc{Certain}_{\mathcal{M}}(Q, I)$?

We are considering the data complexity of the problem.

Computing certain answers (cont'd)

How can $\textsc{Certain}_{\mathcal{M}}(Q, I)$ be computed?

- Naïve algorithm does not work: infinitely many solutions

Computing certain answers (cont'd)

How can $\textsc{Certain}_{\mathcal{M}}(Q, I)$ be computed?

- Naïve algorithm does not work: infinitely many solutions

Approach proposed in [FKMP03]:

1. Materialize a solution J for I such that:

$$\textsc{Certain}_{\mathcal{M}}(Q, I) \quad = \quad Q(J)$$

2. Compute $Q(J)$

Computing certain answers (cont'd)

How can $\textsc{Certain}_{\mathcal{M}}(Q, I)$ be computed?

- Naïve algorithm does not work: infinitely many solutions

Approach proposed in [FKMP03]:

1. Materialize a solution J for I such that:

$$\textsc{Certain}_{\mathcal{M}}(Q, I) \quad = \quad Q(J)$$

2. Compute $Q(J)$

This works well for positive queries!

A solution to materialize: Canonical universal solution

Input: $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ and an instance I of $\mathbf{S}$

Output: Canonical universal solution $\textsc{Can}(I)$ for I

Algorithm:

for every $\varphi(\bar{x}, \bar{y}) \to \exists \bar{z}\, \psi(\bar{x}, \bar{z}) \in \Sigma$ do
  for every $(\bar{a}, \bar{b})$ such that I satisfies $\varphi(\bar{a}, \bar{b})$ do
    create a fresh tuple of null values $\bar{n}$
    insert $\psi(\bar{a}, \bar{n})$ into $\textsc{Can}(I)$

Example: Canonical universal solution

Previous example: $Book(x_1, x_2, y_1) \rightarrow \exists z_1\, Writer(x_2, x_1, z_1)$

I :

Book	*Title*	*AuthorName*	*Affiliation*
	Algebra	Hungerford	U. Washington
	Real Analysis	Royden	Stanford

Example: Canonical universal solution

Previous example: $Book(x_1, x_2, y_1) \to \exists z_1\, Writer(x_2, x_1, z_1)$

I :

Book	*Title*	*AuthorName*	*Affiliation*
	Algebra	Hungerford	U. Washington
	Real Analysis	Royden	Stanford

We have that:

$\textsc{Can}(I)$:

Writer	*Name*	*BookTitle*	*Year*
	Hungerford	Algebra	$\perp_1$
	Royden	Real Analysis	$\perp_2$

Canonical universal solution: Computing certain answers

Canonical universal solution can be computed in polynomial time.

- Data complexity: Data exchange setting is fixed.

Canonical universal solution: Computing certain answers

Canonical universal solution can be computed in polynomial time.

- Data complexity: Data exchange setting is fixed.

Notation: $\mathbf{C}(a)$ holds if and only if a is a constant.

Canonical universal solution: Computing certain answers

Canonical universal solution can be computed in polynomial time.

- Data complexity: Data exchange setting is fixed.

Notation: $\mathbf{C}(a)$ holds if and only if a is a constant.

Theorem (FKMP03)

Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, $Q(x_1, \ldots, x_k)$ a union of conjunctive queries over $\mathbf{T}$ and

$$Q^\star(x_1, \ldots x_k) \quad = \quad \mathbf{C}(x_1) \wedge \cdots \wedge \mathbf{C}(x_k) \wedge Q(x_1, \ldots, x_k).$$

Then for every instance I of $\mathbf{S}$: $\textsc{Certain}_{\mathcal{M}}(Q, I) = Q^\star(\textsc{Can}(I))$.

Why does the previous approach work?

Simple explanation: Closure under homomorphisms

Why does the previous approach work?

Simple explanation: Closure under homomorphisms

$h : \mathrm{dom}(J_1) \to \mathrm{dom}(J_2)$ is a *homomorphism* from J_1 to J_2 if:

- h preserves the relations: If $R(a_1, \ldots, a_k)$ is in J_1, then $R(h(a_1), \ldots, h(a_k))$ is in J_2.
- h is the identity on constants.

A solution J for I under $\mathcal{M}$ is *universal* if:

- For every solution J' for I under $\mathcal{M}$, there exists a homomorphism from J to J'.

$\textsc{Can}(I)$ is a universal solution for I

Why does the previous approach work? (cont'd)

Theorem (FKMP03)

Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, $Q(x_1, \ldots, x_k)$ a union of conjunctive queries over $\mathbf{T}$ and

$$Q^\star(x_1, \ldots x_k) \quad = \quad \mathbf{C}(x_1) \wedge \cdots \wedge \mathbf{C}(x_k) \wedge Q(x_1, \ldots, x_k).$$

Then for every instance I of $\mathbf{S}$ and universal solution J for I under $\mathcal{M}$: $\textsc{Certain}_{\mathcal{M}}(Q, I) = Q^\star(J)$.

Why does the previous approach work? (cont'd)

Theorem (FKMP03)

Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, $Q(x_1, \ldots, x_k)$ a union of conjunctive queries over $\mathbf{T}$ and

$$Q^\star(x_1, \ldots x_k) \quad = \quad \mathbf{C}(x_1) \wedge \cdots \wedge \mathbf{C}(x_k) \wedge Q(x_1, \ldots, x_k).$$

Then for every instance I of $\mathbf{S}$ and universal solution J for I under $\mathcal{M}$: $\textsc{Certain}_{\mathcal{M}}(Q, I) = Q^\star(J)$.

Proof: From the fact that $Q^\star$ is closed under homomorphisms

DATALOG as a query language for data exchange systems

The previous approach works for any language closed under homomorphisms.

- DATALOG queries can also be computed in polynomial time.

DATALOG as a query language for data exchange systems

The previous approach works for any language closed under homomorphisms.

- DATALOG queries can also be computed in polynomial time.

Unfortunately, both DATALOG and union of conjunctive queries keep us on the realm of positive.

Outline

- The data exchange scenario
  - Query answering
- Queries with negation in data exchange
- $\textsc{Datalog}^{\mathbf{C}(\neq)}$ programs
- Beyond union of conjunctive queries
  - Expressive power of $\textsc{Datalog}^{\mathbf{C}(\neq)}$
  - New tractable classes of queries
- Concluding remarks

Are queries with negation interesting in data exchange?

$$\mathcal{M} \quad : \quad \begin{array}{rcl} G(x,y) & \rightarrow & E(x,y) \\ S(x) & \rightarrow & P(x) \\ T(x) & \rightarrow & R(x) \end{array}$$

$$Q \quad : \quad \exists x \exists y \exists z (E(x,z) \wedge E(z,y) \wedge \neg E(x,y))$$

Are queries with negation interesting in data exchange?

$$\mathcal{M} \quad : \quad \begin{array}{rcl} G(x,y) & \rightarrow & E(x,y) \\ S(x) & \rightarrow & P(x) \\ T(x) & \rightarrow & R(x) \end{array}$$

$$Q \quad : \quad \exists x \exists y \exists z (E(x,z) \wedge E(z,y) \wedge \neg E(x,y))$$

$$\mathrm{I} \quad : \quad G(a,b),\ G(b,c)$$

Are queries with negation interesting in data exchange?

$$\mathcal{M} \quad : \quad \begin{array}{rcl} G(x,y) & \rightarrow & E(x,y) \\ S(x) & \rightarrow & P(x) \\ T(x) & \rightarrow & R(x) \end{array}$$

$$Q \quad : \quad \exists x \exists y \exists z (E(x,z) \wedge E(z,y) \wedge \neg E(x,y))$$

$$\begin{array}{lcl} \mathrm{I} & : & G(a,b),\ G(b,c) \\ J_1 & : & E(a,b),\ E(b,c) \end{array}$$

Are queries with negation interesting in data exchange?

$$\mathcal{M} \quad : \quad \begin{array}{rcl} G(x,y) & \rightarrow & E(x,y) \\ S(x) & \rightarrow & P(x) \\ T(x) & \rightarrow & R(x) \end{array}$$

$$Q \quad : \quad \exists x \exists y \exists z (E(x,z) \wedge E(z,y) \wedge \neg E(x,y))$$

$$\begin{array}{lcl} \mathsf{I} & : & G(a,b),\ G(b,c) \\ J_1 & : & E(a,b),\ E(b,c) \\ J_2 & : & E(a,b),\ E(b,c),\ E(a,c) \end{array}$$

Are queries with negation interesting in data exchange?

$$\mathcal{M} \quad : \quad \begin{array}{rcl} G(x,y) & \rightarrow & E(x,y) \\ S(x) & \rightarrow & P(x) \\ T(x) & \rightarrow & R(x) \end{array}$$

$$Q \quad : \quad \exists x \exists y \exists z (E(x,z) \wedge E(z,y) \wedge \neg E(x,y))$$

$$\begin{array}{lcl} \mathrm{I} & : & G(a,b),\ G(b,c) \\ J_1 & : & E(a,b),\ E(b,c) \\ J_2 & : & E(a,b),\ E(b,c),\ E(a,c) \end{array}$$

- Q is always false in a solution where E represents the transitive closure of G

But if positive queries are also considered ...

$$
\begin{array}{lcrcl}
\mathcal{M} & : & G(x,y) & \rightarrow & E(x,y) \\
 & & S(x) & \rightarrow & P(x) \\
 & & T(x) & \rightarrow & R(x) \\
 & & & & \\
Q & : & \multicolumn{3}{l}{\exists x \exists y (P(x) \wedge R(y) \wedge E(x,y)) \vee} \\
 & & \multicolumn{3}{r}{\exists x \exists y \exists z (E(x,z) \wedge E(z,y) \wedge \neg E(x,y))}
\end{array}
$$

But if positive queries are also considered ...

$$\mathcal{M} \quad : \quad \begin{array}{rcl} G(x,y) & \rightarrow & E(x,y) \\ S(x) & \rightarrow & P(x) \\ T(x) & \rightarrow & R(x) \end{array}$$

$$\begin{array}{lll} Q & : & \exists x \exists y (P(x) \wedge R(y) \wedge E(x,y)) \vee \\ & & \qquad \exists x \exists y \exists z (E(x,z) \wedge E(z,y) \wedge \neg E(x,y)) \end{array}$$

$\textsc{Certain}_{\mathcal{M}}(Q, I) = \textit{true}$ iff there exist a, b such that:

- $P(a)$ and $R(b)$ hold in I
- (a, b) is in the transitive closure of G in I

Is there a *reasonable* query language with negation in data exchange?

We cannot directly add inequalities or negated relational atoms to DATALOG.

- Preservation under homomorphisms is lost
- Language becomes intractable, even for conjunctive queries with inequalities (AD98)

Is there a *reasonable* query language with negation in data exchange?

We cannot directly add inequalities or negated relational atoms to DATALOG.

- Preservation under homomorphisms is lost
- Language becomes intractable, even for conjunctive queries with inequalities (AD98)

Homomorphisms in data exchange are the identity on constants

- Inequalities witnessed by constants are preserved under homomorphisms

Our contributions

Query language that extends DATALOG with a form of negation

- As good as DATALOG for data exchange
- Can be used to find new tractable classes of queries

Outline

- The data exchange scenario
  - Query answering
- Queries with negation in data exchange
- $\textsc{Datalog}^{\mathsf{C}(\neq)}$ programs
- Beyond union of conjunctive queries
  - Expressive power of $\textsc{Datalog}^{\mathsf{C}(\neq)}$
  - New tractable classes of queries
- Concluding remarks

$\textsc{Datalog}^{\mathbf{C}(\neq)}$ programs

Definition

A *constant-inequality Datalog rule* is a rule of the form:

$$S(\bar{x}) \leftarrow S_1(\bar{x}_1), \ldots, S_\ell(\bar{x}_\ell), \mathbf{C}(y_1), \ldots, \mathbf{C}(y_m), u_1 \neq v_1, \ldots, u_n \neq v_n$$

where

- S, S_1, ..., S_ℓ are predicate symbols,
- every variable in $\bar{x}$ is mentioned in some tuple $\bar{x}_i$,
- every variable y_j is mentioned in some tuple $\bar{x}_i$, and
- every variable u_j, and every variable v_j, is equal to some variable y_i.

$\textsc{Datalog}^{\mathsf{C}(\neq)}$ programs (cont'd)

A *constant-inequality Datalog program* ($\textsc{Datalog}^{\mathsf{C}(\neq)}$ program) is a finite set of constant-inequality Datalog rules.

$\textsc{Datalog}^{\mathbf{C}(\neq)}$ programs (cont'd)

A *constant-inequality Datalog program* ($\textsc{Datalog}^{\mathbf{C}(\neq)}$ program) is a finite set of constant-inequality Datalog rules.

Example:

$$
\begin{array}{rcl}
S(x,y) & \leftarrow & E(x,y) \\
S(x,y) & \leftarrow & S(x,u), S(u,v), S(v,y), \mathbf{C}(x), \mathbf{C}(u), \mathbf{C}(v), u \neq v
\end{array}
$$

$\textsc{Datalog}^{\mathbf{C}(\neq)}$ programs (cont'd)

A *constant-inequality Datalog program* ($\textsc{Datalog}^{\mathbf{C}(\neq)}$ program) is a finite set of constant-inequality Datalog rules.

Example:

$$
\begin{array}{rcl}
S(x,y) & \leftarrow & E(x,y) \\
S(x,y) & \leftarrow & S(x,u), S(u,v), S(v,y), \mathbf{C}(x), \mathbf{C}(u), \mathbf{C}(v), u \neq v
\end{array}
$$

$\textsc{Datalog}^{\mathbf{C}(\neq)}$ programs (cont'd)

A *constant-inequality Datalog program* ($\textsc{Datalog}^{\mathbf{C}(\neq)}$ program) is a finite set of constant-inequality Datalog rules.

Example:

$$
\begin{array}{rcl}
S(x,y) & \leftarrow & E(x,y) \\
S(x,y) & \leftarrow & S(x,u), S(u,v), S(v,y), \mathbf{C}(x), \mathbf{C}(u), \mathbf{C}(v), u \neq v
\end{array}
$$

$\text{DATALOG}^{\mathsf{C}(\neq)}$ programs can be evaluated efficiently

$\text{DATALOG}^{\mathsf{C}(\neq)}$ programs are preserved under homomorphisms

$\text{Datalog}^{\mathbf{C}(\neq)}$ programs can be evaluated efficiently

$\text{Datalog}^{\mathbf{C}(\neq)}$ programs are preserved under homomorphisms

Proposition

Certain answers of $\text{Datalog}^{\mathbf{C}(\neq)}$ programs can be computed by evaluating the programs over the canonical universal solution.

$\text{DATALOG}^{\mathbf{C}(\neq)}$ programs can be evaluated efficiently

$\text{DATALOG}^{\mathbf{C}(\neq)}$ programs are preserved under homomorphisms

Proposition

Certain answers of $\text{DATALOG}^{\mathbf{C}(\neq)}$ programs can be computed by evaluating the programs over the canonical universal solution.

Theorem

Computing the certain answers of a $\text{DATALOG}^{\mathbf{C}(\neq)}$ *program takes polynomial time (data complexity)*

Outline

- The data exchange scenario
  - Query answering
- Queries with negation in data exchange
- $\textsc{Datalog}^{\mathbf{C}(\neq)}$ programs
- Beyond union of conjunctive queries
  - Expressive power of $\textsc{Datalog}^{\mathbf{C}(\neq)}$
  - New tractable classes of queries
- Concluding remarks

$\textsc{Datalog}^{\mathsf{C}(\neq)}$ programs can express queries with negation

Theorem (ABR09)

For every union of conjunctive queries Q with at most one

- *negated relational atom or*
- *inequality*

per disjunct, there exists a $\textsc{Datalog}^{\mathsf{C}(\neq)}$ program Π such that

$$Q \text{ and } \Pi \text{ are equivalent in the data exchange scenario.}$$

$\textsc{Datalog}^{\mathbf{C}(\neq)}$ programs can express queries with negation

Theorem (ABR09)

For every union of conjunctive queries Q with at most one

- *negated relational atom or*
- *inequality*

per disjunct, there exists a $\textsc{Datalog}^{\mathbf{C}(\neq)}$ program Π such that

for every data exchange setting $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ and instance I of $\mathbf{S}$: $\textsc{Certain}_{\mathcal{M}}(Q, I) = \textsc{Certain}_{\mathcal{M}}(\Pi, I)$.

$\text{DATALOG}^{\mathbf{C}(\neq)}$ programs can express queries with negation

Theorem (ABR09)

For every union of conjunctive queries Q with at most one

- *negated relational atom or*
- *inequality*

per disjunct, there exists a $\text{DATALOG}^{\mathbf{C}(\neq)}$ program Π such that

for every data exchange setting $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ and instance I of $\mathbf{S}$: $\text{CERTAIN}_{\mathcal{M}}(Q, I) = \text{CERTAIN}_{\mathcal{M}}(\Pi, I)$.

- Certain answers for this class of queries can be computed in polynomial time.
  - Π can be computed from Q in polynomial time.
- Result for inequalities was proved in [FKMP03].

Example: Expressing negation in $\text{DATALOG}^{\mathbf{C}(\neq)}$

$$Q \;:\; \exists x \exists y\ (E(x,y) \wedge x \neq y) \vee \exists x \exists y \exists z\ (E(x,y) \wedge E(y,z) \wedge \neg E(x,z))$$

Example: Expressing negation in $\text{DATALOG}^{\mathbf{C}(\neq)}$

$$Q \;:\; \exists x \exists y\ (E(x,y) \wedge x \neq y) \vee \\ \exists x \exists y \exists z\ (E(x,y) \wedge E(y,z) \wedge \neg E(x,z))$$

$$\begin{aligned} dom(x) &\leftarrow E(x,z) \\ dom(x) &\leftarrow E(z,x) \end{aligned}$$

Collect the domain

Example: Expressing negation in $\text{Datalog}^{\mathbf{C}(\neq)}$

$$Q \;:\; \exists x \exists y\, (E(x,y) \wedge x \neq y) \vee \exists x \exists y \exists z\, (E(x,y) \wedge E(y,z) \wedge \neg E(x,z))$$

$$\begin{array}{rcl}
dom(x) & \leftarrow & E(x,z) \\
dom(x) & \leftarrow & E(z,x) \\
EQ(x,x) & \leftarrow & dom(x) \\
EQ(x,y) & \leftarrow & EQ(y,x) \\
EQ(x,y) & \leftarrow & EQ(x,z), EQ(z,y)
\end{array}$$

Formalize equality

Example: Expressing negation in $\text{DATALOG}^{\mathbf{C}(\neq)}$

$$
\begin{aligned}
Q \;:\; & \exists x \exists y \,(E(x,y) \wedge x \neq y) \vee \\
& \qquad \exists x \exists y \exists z \,(E(x,y) \wedge E(y,z) \wedge \neg E(x,z))
\end{aligned}
$$

$$
\begin{array}{rcl}
dom(x) & \leftarrow & E(x,z) \\
dom(x) & \leftarrow & E(z,x) \\
EQ(x,x) & \leftarrow & dom(x) \\
EQ(x,y) & \leftarrow & EQ(y,x) \\
EQ(x,y) & \leftarrow & EQ(x,z), EQ(z,y) \\
U(x,y) & \leftarrow & E(x,y)
\end{array}
$$

Copy E into U

Example: Expressing negation in $\text{DATALOG}^{\mathbf{C}(\neq)}$

$$Q \; : \; \exists x \exists y \; (E(x,y) \wedge x \neq y) \vee \exists x \exists y \exists z \; (E(x,y) \wedge E(y,z) \wedge \neg E(x,z))$$

$$\begin{array}{rcl}
dom(x) & \leftarrow & E(x,z) \\
dom(x) & \leftarrow & E(z,x) \\
EQ(x,x) & \leftarrow & dom(x) \\
EQ(x,y) & \leftarrow & EQ(y,x) \\
EQ(x,y) & \leftarrow & EQ(x,z), EQ(z,y) \\
U(x,y) & \leftarrow & E(x,y) \\
U(x,y) & \leftarrow & U(u,v), EQ(u,x), EQ(v,y)
\end{array}$$

Replace equals in U

Example: Expressing negation in $\text{DATALOG}^{\mathbf{C}(\neq)}$

$$Q \;:\; \exists x \exists y\ (E(x,y) \wedge x \neq y) \vee \exists x \exists y \exists z\ (E(x,y) \wedge E(y,z) \wedge \neg E(x,z))$$

Simulate 2nd disjunct of Q

$$\begin{array}{rcl}
dom(x) & \leftarrow & E(x,z) \\
dom(x) & \leftarrow & E(z,x) \\
EQ(x,x) & \leftarrow & dom(x) \\
EQ(x,y) & \leftarrow & EQ(y,x) \\
EQ(x,y) & \leftarrow & EQ(x,z), EQ(z,y) \\
U(x,y) & \leftarrow & E(x,y) \\
U(x,y) & \leftarrow & U(u,v), EQ(u,x), EQ(v,y) \\
U(x,y) & \leftarrow & U(x,z), U(z,y)
\end{array}$$

Example: Expressing negation in $\text{DATALOG}^{\mathsf{C}(\neq)}$

$$Q \; : \; \exists x \exists y \; (E(x,y) \wedge x \neq y) \vee \\ \exists x \exists y \exists z \; (E(x,y) \wedge E(y,z) \wedge \neg E(x,z))$$

Simulate 1st disjunct of Q

$$\begin{array}{rcl}
dom(x) & \leftarrow & E(x,z) \\
dom(x) & \leftarrow & E(z,x) \\
EQ(x,x) & \leftarrow & dom(x) \\
EQ(x,y) & \leftarrow & EQ(y,x) \\
EQ(x,y) & \leftarrow & EQ(x,z), EQ(z,y) \\
U(x,y) & \leftarrow & E(x,y) \\
U(x,y) & \leftarrow & U(u,v), EQ(u,x), EQ(v,y) \\
U(x,y) & \leftarrow & U(x,z), U(z,y) \\
EQ(x,y) & \leftarrow & U(x,y)
\end{array}$$

Example: Expressing negation in DATALOG$^{\mathbf{C}(\neq)}$

$$Q \;:\; \exists x \exists y\ (E(x,y) \wedge x \neq y) \vee \\ \exists x \exists y \exists z\ (E(x,y) \wedge E(y,z) \wedge \neg E(x,z))$$

$$\begin{array}{rcl}
dom(x) & \leftarrow & E(x,z) \\
dom(x) & \leftarrow & E(z,x) \\
EQ(x,x) & \leftarrow & dom(x) \\
EQ(x,y) & \leftarrow & EQ(y,x) \\
EQ(x,y) & \leftarrow & EQ(x,z), EQ(z,y) \\
U(x,y) & \leftarrow & E(x,y) \\
U(x,y) & \leftarrow & U(u,v), EQ(u,x), EQ(v,y) \\
U(x,y) & \leftarrow & U(x,z), U(z,y) \\
EQ(x,y) & \leftarrow & U(x,y) \\
\text{TRUE} & \leftarrow & EQ(x,y), \mathbf{C}(x), \mathbf{C}(y), x \neq y
\end{array}$$

Answer Q

Outline

- The data exchange scenario
  - Query answering
- Queries with negation in data exchange
- $\textsc{Datalog}^{\mathbf{C}(\neq)}$ programs
- Beyond union of conjunctive queries
  - Expressive power of $\textsc{Datalog}^{\mathbf{C}(\neq)}$
  - New tractable classes of queries
- Concluding remarks

Certain answers for conjunctive queries with two inequalities

Theorem (M05)

The certain answers problem is coNP*-complete for the class of conjunctive queries with two inequalities (data complexity).*

Certain answers for conjunctive queries with two inequalities

Theorem (M05)

The certain answers problem is coNP*-complete for the class of conjunctive queries with two inequalities (data complexity).*

An interesting tractable fragment of $2\text{-}\mathrm{CQ}^{\neq}$ was identified by considering $\textsc{Datalog}^{\mathbf{C}(\neq)}$ programs.

Two restrictions on queries are needed

Just the intuition:

Two restrictions on queries are needed

Just the intuition:

- Constant Joins: Null values do not witness a join

Two restrictions on queries are needed

Just the intuition:

- Constant Joins: Null values do not witness a join
- Almost constant inequalities: Every inequality is not witnessed just by null values

A tractable fragment of 2-$\mathrm{UCQ}^{\neq}$

Theorem (ABR09)

For every 2-$\mathrm{UCQ}^{\neq}$ Q with:

- *constant joins and*
- *almost constant inequalities,*

there exists a $\mathrm{D{\scriptsize ATALOG}}^{\mathsf{C}(\neq)}$ program Π such that:

Q and Π are equivalent in the data exchange scenario.

A tractable fragment of 2-$\mathrm{UCQ}^{\neq}$

Theorem (ABR09)

For every 2-$\mathrm{UCQ}^{\neq}$ Q with:

- *constant joins and*
- *almost constant inequalities,*

there exists a $\textsc{Datalog}^{\mathbf{C}(\neq)}$ *program* Π *such that:*

for every data exchange setting $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ *and instance* I *of* $\mathbf{S}$*:* $\textsc{Certain}_{\mathcal{M}}(Q, I) = \textsc{Certain}_{\mathcal{M}}(\Pi, I)$*.*

A tractable fragment of 2-$\mathrm{UCQ}^{\neq}$

Theorem (ABR09)

For every 2-$\mathrm{UCQ}^{\neq}$ Q with:

- *constant joins and*
- *almost constant inequalities,*

there exists a $\textsc{Datalog}^{\mathbf{C}(\neq)}$ *program* Π *such that:*

for every data exchange setting $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ *and instance* I *of* $\mathbf{S}$: $\textsc{Certain}_{\mathcal{M}}(Q, I) = \textsc{Certain}_{\mathcal{M}}(\Pi, I)$.

Certain answers to this class of queries can be computed in polynomial time

- Π can be computed from Q in polynomial time.

A tractable fragment of 2-$\mathrm{UCQ}^{\neq}$ (cont'd)

Theorem (ABR09)

Removing any one of the conditions in the previous theorem yields to coNP*-completeness.*

A tractable fragment of 2-$\mathrm{UCQ}^{\neq}$ (cont'd)

Theorem (ABR09)

Removing any one of the conditions in the previous theorem yields to CONP*-completeness.*

This result is stronger than the result in [M05].

- Reduction in [M05] does not impose these restrictions.

Outline

- The data exchange scenario
  - Query answering
- Queries with negation in data exchange
- $\text{DATALOG}^{\mathbf{C}(\neq)}$ programs
- Beyond union of conjunctive queries
  - Expressive power of $\text{DATALOG}^{\mathbf{C}(\neq)}$
  - New tractable classes of queries
- Concluding remarks

We propose $\textsc{Datalog}^{\mathsf{C}(\neq)}$ as a query language for data exchange systems

- Certain answers to a $\textsc{Datalog}^{\mathsf{C}(\neq)}$ program can be computed in polynomial time (data complexity).
- $\textsc{Datalog}^{\mathsf{C}(\neq)}$ is equipped with a form of negation.
  - Union of conjunctive queries with one negated atom per disjunct are expressible in $\textsc{Datalog}^{\mathsf{C}(\neq)}$.
- $\textsc{Datalog}^{\mathsf{C}(\neq)}$ can be used to find new tractable classes of queries with negation.
  - In this talk: A fragment of 2-$\mathrm{UCQ}^{\neq}$

Thank you!

Two restrictions on queries are needed

Two restrictions on queries are needed

Constant Joins

Null values do not witness a join

$$\mathcal{M} \quad : \quad \begin{array}{rcl} P(u,v) & \rightarrow & T(u,v) \\ Q(u,v) & \rightarrow & \exists w\, U(u,w) \end{array}$$

$$Q_1 \quad : \quad \exists x \exists y \exists z\, (T(x,y) \land U(x,z))$$

$$Q_2 \quad : \quad \exists x \exists y \exists z\, (U(x,z) \land U(y,z))$$

Two restrictions on queries are needed

Constant Joins

Null values do not witness a join

$$\mathcal{M} \quad : \quad \begin{array}{rcl} P(u,v) & \rightarrow & T(u,v) \\ Q(u,v) & \rightarrow & \exists w\, U(u,w) \end{array}$$

$Q_1 \quad : \quad \exists x \exists y \exists z\, (T(x,y) \wedge U(x,z))$ YES

$Q_2 \quad : \quad \exists x \exists y \exists z\, (U(x,z) \wedge U(y,z))$

Two restrictions on queries are needed

Constant Joins

Null values do not witness a join

$$\mathcal{M} \;:\; \begin{array}{rcl} P(u,v) & \rightarrow & T(u,v) \\ Q(u,v) & \rightarrow & \exists w\, U(u,w) \end{array}$$

Q_1 : $\exists x \exists y \exists z\, (T(x,y) \wedge U(x,z))$ YES

Q_2 : $\exists x \exists y \exists z\, (U(x,z) \wedge U(y,z))$ NO

Two restrictions on queries are needed

Almost constant inequalities

Every inequality is not witnessed just by null values

$$\mathcal{M} \quad : \quad \begin{array}{rcl} P(u,v) & \to & T(u,v) \\ Q(u,v) & \to & \exists w\, U(u,w) \end{array}$$

$$Q_1 \quad : \quad \exists x \exists y \exists z\, (U(x,y) \wedge U(x,z) \wedge x \neq z)$$

$$Q_2 \quad : \quad \exists x \exists y \exists z\, (U(x,y) \wedge U(x,z) \wedge y \neq z)$$

Two restrictions on queries are needed

Almost constant inequalities

Every inequality is not witnessed just by null values

$$\mathcal{M} \quad : \quad \begin{array}{rcl} P(u,v) & \rightarrow & T(u,v) \\ Q(u,v) & \rightarrow & \exists w\, U(u,w) \end{array}$$

$Q_1 \quad : \quad \exists x \exists y \exists z\, (U(x,y) \wedge U(x,z) \wedge x \neq z)$ YES

$Q_2 \quad : \quad \exists x \exists y \exists z\, (U(x,y) \wedge U(x,z) \wedge y \neq z)$

Two restrictions on queries are needed

Almost constant inequalities

Every inequality is not witnessed just by null values

$$\mathcal{M} \;:\; \begin{array}{rcl} P(u,v) & \rightarrow & T(u,v) \\ Q(u,v) & \rightarrow & \exists w\, U(u,w) \end{array}$$

$Q_1 \;:\; \exists x \exists y \exists z\, (U(x,y) \wedge U(x,z) \wedge x \neq z)$ YES

$Q_2 \;:\; \exists x \exists y \exists z\, (U(x,y) \wedge U(x,z) \wedge y \neq z)$ NO