Data Exchange:

Source instance $\Rightarrow$ Target instance

Data Exchange:
Source instance $\Rightarrow$ Target instance

Problem 1:

- There may be infinitely many valid target instances for a given source instance

Data Exchange:
Source instance $\Rightarrow$ Target instance

Problem 1:

- There may be infinitely many valid target instances for a given source instance

Problem 2. Query Answering

Data Exchange:
Source instance $\Rightarrow$ Target instance

Problem 1:

- There may be infinitely many valid target instances for a given source instance

Problem 2. Query Answering

- What does it mean to answer a query over the target schema?
- Can we answer queries using only one target instance?

Data Exchange:
Source instance $\Rightarrow$ Target instance

Problem 1:

- There may be infinitely many valid target instances for a given source instance

Problem 2. Query Answering

- What does it mean to answer a query over the target schema?
- Can we answer queries using only one target instance?

Fagin, Kolaitis, Miller, Popa, 2003:

- Use a certain answers semantics
- Canonical Solution: "good" target instance that can be computed in polynomial time
- Union of conjunctive queries: their certain answers can be computed using only the canonical solution

We propose a tractable query language that express *negation*

For union of conjunctive queries, the certain answers can be computed in polynomial time.

- Union of conjunctive queries have this good property because they are preserved under homomorphisms

We propose a tractable query language that express *negation*

For union of conjunctive queries, the certain answers can be computed in polynomial time.

- Union of conjunctive queries have this good property because they are preserved under homomorphisms
- DATALOG queries can also be computed in polynomial time

We propose a tractable query language that express *negation*

For union of conjunctive queries, the certain answers can be computed in polynomial time.

- Union of conjunctive queries have this good property because they are preserved under homomorphisms
- DATALOG queries can also be computed in polynomial time
- Both DATALOG and union of conjunctive queries keep us on the realm of positive

We propose a tractable query language that express *negation*

For union of conjunctive queries, the certain answers can be computed in polynomial time.

- Union of conjunctive queries have this good property because they are preserved under homomorphisms
- DATALOG queries can also be computed in polynomial time
- Both DATALOG and union of conjunctive queries keep us on the realm of positive

- Computing certain answers of conjunctive queries with inequalities is coNP-complete

We propose a tractable query language that express *negation*

For union of conjunctive queries, the certain answers can be computed in polynomial time.

- Union of conjunctive queries have this good property because they are preserved under homomorphisms
- DATALOG queries can also be computed in polynomial time
- Both DATALOG and union of conjunctive queries keep us on the realm of positive

- Computing certain answers of conjunctive queries with inequalities is coNP-complete

How can we add negation while keeping good properties for data exchange?

Query Languages for Data Exchange: Beyond Unions of Conjunctive Queries

Marcelo Arenas
PUC Chile

Pablo Barceló
Univ. of Chile

Juan Reutter
PUC Chile

Khipu: South Andean Center for Database Research

A bit of notation...

Data exchange settings:

- Source schema $\mathbf{S}$ (Source instances with constant values)
- Target schema $\mathbf{T}$ (Target instance can contain nulls)
- Set Σ_{st} of *st-tgds* of the form:

$$\phi_{\mathbf{S}}(\bar{x}) \rightarrow \exists \bar{y} \psi_{\mathbf{T}}(\bar{x}, \bar{y})$$

- $\mathbf{C}(a)$ holds if a is a constant value

An instance J is a *solution* for I if

- $(I, J) \models \Sigma_{st}$

Homomorphism and Universal Solutions

A *homomorphism* from J_1 to J_2 is a function that:

- Preserve the relations
- Is the identity on constants

J is a *universal* solution if

- There is a homomorphism from J to every other solution

Homomorphism and Universal Solutions

A *homomorphism* from J_1 to J_2 is a function that:

- Preserve the relations
- Is the identity on constants

J is a *universal* solution if

- There is a homomorphism from J to every other solution

Canonical universal solution can be computed in polynomial time using a chase procedure (FKMP 03).

Certain answers for conjunctive queries with negation are empty/false

Example:

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

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

Certain answers for conjunctive queries with negation are empty/false

Example:

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

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

J_1 :

$E(a,b)$
$E(b,c)$

Certain answers for conjunctive queries with negation are empty/false

Example:

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

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

J_2 :

$E(a,b)$

$E(b,c)$

$E(a,c)$

Certain answers for conjunctive queries with negation are empty/false

Example:

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

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

J_2 :

$E(a,b)$
$E(b,c)$
$E(a,c)$

J_2 is also a solution!

Certain answers for conjunctive queries with negation are empty/false

Example:

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

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

J_2 :

$E(a,b)$
$E(b,c)$
$E(a,c)$

J_2 is also a solution!

- Idea: solution where E contains the transitive closure of G

Certain answers for conjunctive queries with negation are empty/false

Example:

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

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

J_2 :

$E(a,b)$
$E(b,c)$
$E(a,c)$

J_2 is also a solution!

- Idea: solution where E contains the transitive closure of G
- Q is always false in that solution!

Unions of *positive* queries and conjunctive queries with negation are much more interesting

Example:

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

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

Unions of *positive* queries and conjunctive queries with negation are much more interesting

Example:

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

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

- If we try to falsify the second disjunct (computing the transitive closure of G), we may end up satisfying the first one.

Unions of *positive* queries and conjunctive queries with negation are much more interesting

Example:

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

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

- If we try to falsify the second disjunct (computing the transitive closure of G), we may end up satisfying the first one.
- Q holds if there exist a, b:
  - $P(a)$, $R(b)$ hold
  - (a, b) is in the transitive closure of G

Using DATALOG we compute certain answers for queries with negation in polynomial time

Idea: Encode Q using DATALOG programs

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

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

Using DATALOG we compute certain answers for queries with negation in polynomial time

Idea: Encode Q using DATALOG programs

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

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

$$
\begin{aligned}
S(x,y) &\leftarrow E(x,y) \\
S(x,y) &\leftarrow S(x,z),\ S(z,y) \\
\texttt{true} &\leftarrow P(x),\ R(y),\ S(x,y)
\end{aligned}
$$

Using Datalog we compute certain answers for queries with negation in polynomial time

Idea: Encode Q using Datalog programs

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

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

$$\begin{array}{rcl} S(x,y) & \leftarrow & E(x,y) \\ S(x,y) & \leftarrow & S(x,z),\ S(z,y) \\ \texttt{true} & \leftarrow & P(x),\ R(y),\ S(x,y) \end{array}$$

We only evaluate this program in the canonical solution

Queries with inequalities cannot be answered directly in universal solutions

Problem:
We cannot add inequalities directly to DATALOG.

- Preservation under homomorphisms is lost
- Language becomes intractable (Abiteboul, Dushka 1998)

Homomorphisms in data exchange are the identity on constants

- Thus, inequalities witnessed by constants are preserved under homomorphisms

Contributions

Query Language that extends DATALOG with negation

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

...And further

- Combined complexity of the new language and related query languages

Outline

Formalization

- $\text{DATALOG}^{\mathsf{C}(\neq)}$ programs

Beyond union of conjunctive queries

- Expressive power of $\text{DATALOG}^{\mathsf{C}(\neq)}$
- New tractable classes of queries

Combined Complexity

- $\text{DATALOG}^{\mathsf{C}(\neq)}$ and queries with inequalities
- Restricting to LAV settings

Concluding remarks

$\text{DATALOG}^{\mathbf{C}(\neq)}$ programs extend DATALOG with inequalities over constants

Definition:
A collection of constant-inequality rules of the form:

$S(\bar{x}) \leftarrow \ldots$

- predicate symbols
- variables under predicate **C**
- inequalities of the form $u \neq v$,
  u and v must be under predicate **C**

$\text{D}\scriptstyle\text{ATALOG}^{\mathbf{C}(\neq)}$ programs extend $\text{D}\scriptstyle\text{ATALOG}$ with inequalities over constants

Definition:
A collection of constant-inequality rules of the form:

$S(\bar{x}) \leftarrow \ldots$

- predicate symbols
- variables under predicate $\mathbf{C}$
- inequalities of the form $u \neq v$,
  u and v must be under predicate $\mathbf{C}$

Example:

$$
\begin{array}{rcl}
S(x,y) & \leftarrow & E(x,y) \\
S(x,y) & \leftarrow & S(x,z), S(z,y), \mathbf{C}(x), \mathbf{C}(z), \mathbf{C}(y), x \neq z, y \neq z \\
\mathit{true} & \leftarrow & P(x), R(y), S(x,y), \mathbf{C}(x), \mathbf{C}(y), x \neq y
\end{array}
$$

$\text{DATALOG}^{\mathbf{C}(\neq)}$ programs extend DATALOG with inequalities over constants

Definition:
A collection of constant-inequality rules of the form:

$S(\bar{x}) \leftarrow \ldots$

- predicate symbols
- variables under predicate $\mathbf{C}$
- inequalities of the form $u \neq v$,
  u and v must be under predicate $\mathbf{C}$

Example:

$$
\begin{aligned}
S(x,y) &\leftarrow E(x,y) \\
S(x,y) &\leftarrow S(x,z), S(z,y), \mathbf{C}(x), \mathbf{C}(z), \mathbf{C}(y), x \neq z, y \neq z \\
\mathit{true} &\leftarrow P(x), R(y), S(x,y), \mathbf{C}(x), \mathbf{C}(y), x \neq y
\end{aligned}
$$

$\textsc{Datalog}^{\mathbf{C}(\neq)}$ programs extend $\textsc{Datalog}$ with inequalities over constants

Definition:
A collection of constant-inequality rules of the form:

$S(\bar{x}) \leftarrow \ldots$

- predicate symbols
- variables under predicate $\mathbf{C}$
- inequalities of the form $u \neq v$,
  u and v must be under predicate $\mathbf{C}$

Example:

$$
\begin{aligned}
S(x,y) &\leftarrow E(x,y)\\
S(x,y) &\leftarrow S(x,z), S(z,y), \mathbf{C}(x), \mathbf{C}(z), \mathbf{C}(y), x \neq z, y \neq z\\
\mathit{true} &\leftarrow P(x), R(y), S(x,y), \mathbf{C}(x), \mathbf{C}(y), x \neq y
\end{aligned}
$$

$\textsc{Datalog}^{\mathbf{C}(\neq)}$ programs extend $\textsc{Datalog}$ with inequalities over constants

Definition:
A collection of constant-inequality rules of the form:

$S(\bar{x}) \leftarrow \ldots$

- predicate symbols
- variables under predicate $\mathbf{C}$
- inequalities of the form $u \neq v$,
  u and v must be under predicate $\mathbf{C}$

Example:

$$\begin{array}{rcl} S(x,y) & \leftarrow & E(x,y) \\ S(x,y) & \leftarrow & S(x,z), S(z,y), \mathbf{C}(x), \mathbf{C}(z), \mathbf{C}(y), x \neq z, y \neq z \\ \mathit{true} & \leftarrow & P(x), R(y), S(x,y), \mathbf{C}(x), \mathbf{C}(y), x \neq y \end{array}$$

$\text{DATALOG}^{\mathsf{C}(\neq)}$ programs have the same good properties as conjunctive queries

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

$\text{DATALOG}^{\mathsf{C}(\neq)}$ programs have the same good properties as conjunctive queries

- $\text{DATALOG}^{\mathsf{C}(\neq)}$ programs are preserved under homomorphisms
  - DATALOG programs are preserved under homomorphisms
  - every inequality must be witnessed by constants
  - homomorphisms are the identity on constants

$\text{DATALOG}^{\mathbf{C}(\neq)}$ programs have the same good properties as conjunctive queries

- $\text{DATALOG}^{\mathbf{C}(\neq)}$ programs are preserved under homomorphisms
  - DATALOG programs are preserved under homomorphisms
  - every inequality must be witnessed by constants
  - homomorphisms are the identity on constants

Proposition

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

$\textsc{Datalog}^{\mathbf{C}(\neq)}$ programs have the same good properties as conjunctive queries

- $\textsc{Datalog}^{\mathbf{C}(\neq)}$ programs are preserved under homomorphisms
  - $\textsc{Datalog}$ programs are preserved under homomorphisms
  - every inequality must be witnessed by constants
  - homomorphisms are the identity on constants

Proposition

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

Theorem

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

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

Theorem

Every union of conjunctive query with at most

- One negated atom
- One inequality

per disjunct, can be expressed as a $\textsc{Datalog}^{\mathsf{C}(\neq)}$ program.

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

Theorem

Every union of conjunctive query with at most

- One negated atom
- One inequality

per disjunct, can be expressed as a $\textsc{Datalog}^{\mathsf{C}(\neq)}$ program.

- Certain answers for this class of queries can be computed in polynomial time
- Result for inequalities had been proved by FKMP03 using different techniques

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

Theorem

Every union of conjunctive query with at most

- One negated atom
- One inequality

per disjunct, can be expressed as a $\text{DATALOG}^{\mathsf{C}(\neq)}$ program.

- Certain answers for this class of queries can be computed in polynomial time
- Result for inequalities had been proved by FKMP03 using different techniques
- Next example gives a hint on the proof

Writing $\text{DATALOG}^{\mathbf{C}(\neq)}$ programs to answer queries with negation

$$Q: \quad \exists x \exists y\ (E(x,y) \land x \neq y) \lor$$
$$\exists x \exists y \exists z\ (E(x,y) \land E(y,z) \land \neg E(x,z))$$

Writing $\textsc{Datalog}^{\mathbf{C}(\neq)}$ programs to answer queries with negation

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

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

- Collect the domain

Writing $\textsc{Datalog}^{\mathbf{C}(\neq)}$ programs to answer queries with negation

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

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

- Collect the domain
- Formalize the Equality

Writing DATALOG$^{\mathsf{C}(\neq)}$ programs to answer queries with negation

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

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

- Collect the domain
- Formalize the Equality
- Copy E into U

Writing $\textsc{Datalog}^{\mathbf{C}(\neq)}$ programs to answer queries with negation

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

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

- Collect the domain
- Formalize the Equality
- Copy E into U
- Replace equals in U

Writing $\text{DATALOG}^{\mathbf{C}(\neq)}$ programs to answer queries with negation

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

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

- Collect the domain
- Formalize the Equality
- Copy E into U
- Replace equals in U
- Simulate negation

Writing $\text{DATALOG}^{\mathbf{C}(\neq)}$ programs to answer queries with negation

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

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

- Collect the domain
- Formalize the Equality
- Copy E into U
- Replace equals in U
- Simulate negation
- Simulate inequality

Writing $\text{DATALOG}^{\mathbf{C}(\neq)}$ programs to answer queries with negation

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

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

- Collect the domain
- Formalize the Equality
- Copy E into U
- Replace equals in U
- Simulate negation
- Simulate inequality
- Answer

Outline

Formalization

- $\text{Datalog}^{\mathsf{C}(\neq)}$ programs

Beyond union of conjunctive queries

- Expressive power of $\text{Datalog}^{\mathsf{C}(\neq)}$
- New tractable classes of queries

Combined Complexity

- $\text{Datalog}^{\mathsf{C}(\neq)}$ and queries with inequalities
- Restricting to LAV settings

Concluding remarks

Classes of queries

$(\mathrm{UCQ})\mathrm{CQ}$

- (union) of conjunctive queries

$(\mathrm{UCQ}^{\neq})\mathrm{CQ}^{\neq}$

- (union) of conjunctive queries with inequalities

k-$\mathrm{CQ}^{\neq}$

- conjunctive queries with at most k inequalities

Certain answers for conjunctive queries with two inequalities is intractable (data complexity)

[Madry 05]:

- The certain answers problem is coNP-complete for 2-$\text{CQ}^{\neq}$

Certain answers for conjunctive queries with two inequalities is intractable (data complexity)

[Madry 05]:

- The certain answers problem is coNP-complete for 2-$\mathrm{CQ}^{\neq}$

We find an interesting tractable fragment for this class of queries, using translation into $\textsc{Datalog}^{\mathsf{C}(\neq)}$ programs

We need to define two restrictions

- Constant Joins
- Almost constant inequalities

We need to define two restrictions

- Constant Joins
- Almost constant inequalities

Constant Joins:
No null values can witness a join of a relation

We need to define two restrictions

- Constant Joins
- Almost constant inequalities

Constant Joins:
No null values can witness a join of a relation

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

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

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

We need to define two restrictions

- Constant Joins
- Almost constant inequalities

Constant Joins:
No null values can witness a join of a relation

$\mathcal{M}$: $P(u, v) \rightarrow T(u, v)$

$Q(u, v) \rightarrow \exists w U(u, w)$

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

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

We need to define two restrictions

- Constant Joins
- Almost constant inequalities

Constant Joins:
No null values can witness a join of a relation

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

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

We need to define two restrictions

- Constant Joins
- Almost constant inequalities

Almost constant inequalities:
Every inequality can be witnessed by at most 1 null value

We need to define two restrictions

- Constant Joins
- Almost constant inequalities

Almost constant inequalities:
Every inequality can be witnessed by at most 1 null value

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

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

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

We need to define two restrictions

- Constant Joins
- Almost constant inequalities

Almost constant inequalities:
Every inequality can be witnessed by at most 1 null value

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

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

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

We need to define two restrictions

- Constant Joins
- Almost constant inequalities

Almost constant inequalities:
Every inequality can be witnessed by at most 1 null value

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

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

We use $\textsc{Datalog}^{\mathsf{C}(\neq)}$ to find a tractable fragment for union of conjunctive queries with at most two inequalities

Theorem

Every 2-$\mathrm{UCQ}^{\neq}$ with:

- constant joins
- almost constant inequalities

can be expressed as a $\textsc{Datalog}^{\mathsf{C}(\neq)}$ program in data exchange.

We use $\textsc{Datalog}^{\mathbf{C}(\neq)}$ to find a tractable fragment for union of conjunctive queries with at most two inequalities

Theorem

Every 2-$\mathrm{UCQ}^{\neq}$ with:

- constant joins
- almost constant inequalities

can be expressed as a $\textsc{Datalog}^{\mathbf{C}(\neq)}$ program in data exchange.

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

We use $\textsc{Datalog}^{\mathbf{C}(\neq)}$ to find a tractable fragment for union of conjunctive queries with at most two inequalities

Theorem

Every 2-$\mathrm{UCQ}^{\neq}$ with:

- constant joins
- almost constant inequalities

can be expressed as a $\textsc{Datalog}^{\mathbf{C}(\neq)}$ program in data exchange.

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

- Removing any one of this conditions yields to intractability

We use $\text{DATALOG}^{\mathbf{C}(\neq)}$ to find a tractable fragment for union of conjunctive queries with at most two inequalities

Theorem

Every 2-$\mathrm{UCQ}^{\neq}$ with:

- constant joins
- almost constant inequalities

can be expressed as a $\text{DATALOG}^{\mathbf{C}(\neq)}$ program in data exchange.

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

- Removing any one of this conditions yields to intractability
- Stronger that Madry's proof (did not have these restrictions)

There is no hope for 3-$\mathrm{CQ}^{\neq}$

Theorem

There exists a query Q in 3-$\mathrm{CQ}^{\neq}$ with

- constant joins
- almost constant inequalities

such that computing it's certain answers is coNP-complete.

Outline

Formalization

- $\text{DATALOG}^{\mathbf{C}(\neq)}$ programs

Beyond union of conjunctive queries

- Expressive power of $\text{DATALOG}^{\mathbf{C}(\neq)}$
- New tractable classes of queries

Combined Complexity

- $\text{DATALOG}^{\mathbf{C}(\neq)}$ and queries with inequalities
- Restricting to LAV settings

Concluding remarks

Combined Complexity: a natural question

What is the complexity if we consider as inputs

- Database instance ?

Combined Complexity: a natural question

What is the complexity if we consider as inputs

- Database instance
- Data exchange setting, query ?

Combined Complexity: a natural question

What is the complexity if we consider as inputs

- Database instance
- Data exchange setting, query ?

Kolaitis, Pantajja, Tan 06:

- Combined complexity of existence of solutions
- Lower bounds for query answering: 1-UCQ

Combined Complexity: a natural question

What is the complexity if we consider as inputs

- Database instance
- Data exchange setting, query ?

Kolaitis, Pantajja, Tan 06:

- Combined complexity of existence of solutions
- Lower bounds for query answering: 1-UCQ

We study the combined complexity of query answering

- Tight lower bounds (single conjunctive queries)
- Results for $\textsc{Datalog}^{\mathsf{C}(\neq)}$ and related query languages

Combined Complexity for the general setting

Theorem

Input: Data exchange setting $\mathcal{M}$, query Q, instance I and tuple $\bar{t}$
Problem: Is $\bar{t}$ in the certain answers of Q for I under $\mathcal{M}$?

$\textsc{Exptime}$-complete for $\textsc{Datalog}^{\mathsf{C}(\neq)}$ programs

Combined Complexity for the general setting

Theorem

Input: Data exchange setting $\mathcal{M}$, query Q, instance I and tuple $\bar{t}$
Problem: Is $\bar{t}$ in the certain answers of Q for I under $\mathcal{M}$?

$\textsc{Exptime}$-complete for $\textsc{Datalog}^{\mathbf{C}(\neq)}$ programs
$\textsc{Exptime}$-complete for 1-$\mathrm{CQ}^{\neq}$

Combined Complexity for the general setting

Theorem

Input: Data exchange setting $\mathcal{M}$, query Q, instance I and tuple $\bar{t}$
Problem: Is $\bar{t}$ in the certain answers of Q for I under $\mathcal{M}$?

$\textsc{Exptime}$-complete	for $\textsc{Datalog}^{\mathbf{C}(\neq)}$ programs
$\textsc{Exptime}$-complete	for 1-$\mathrm{CQ}^{\neq}$
$\textsc{coNExptime}$-complete	for k-$\mathrm{CQ}^{\neq}$, $k \geq 2$
$\textsc{coNExptime}$-complete	for $\mathrm{CQ}^{\neq}$

Combined Complexity for the general setting

Theorem

Input: Data exchange setting $\mathcal{M}$, query Q, instance I and tuple $\bar{t}$
Problem: Is $\bar{t}$ in the certain answers of Q for I under $\mathcal{M}$?

$\textsc{Exptime}$-complete	for $\textsc{Datalog}^{\mathbf{C}(\neq)}$ programs
$\textsc{Exptime}$-complete	for 1-$\mathrm{CQ}^{\neq}$
$\textsc{coNexptime}$-complete	for k-$\mathrm{CQ}^{\neq}$, $k \geq 2$
$\textsc{coNexptime}$-complete	for $\mathrm{CQ}^{\neq}$

- Same results hold for unions
- It follows from KPT06 that the problem is $\textsc{Exptime}$-complete for 1-$\mathrm{UCQ}^{\neq}$

Lower combined complexity if we restrict to LAV settings

A LAV setting is a data exchange settings where Σ_{st} is of the form:

$$R(\bar{x}) \rightarrow \exists \bar{y} \psi(\bar{x}, \bar{y})$$

- Premises are single relational atoms

Lower combined complexity if we restrict to LAV settings

A LAV setting is a data exchange settings where Σ_{st} is of the form:

$$R(\bar{x}) \rightarrow \exists \bar{y} \psi(\bar{x}, \bar{y})$$

- Premises are single relational atoms

Very used in practice!

Lower combined complexity if we restrict to LAV settings

A LAV setting is a data exchange settings where Σ_{st} is of the form:

$$R(\bar{x}) \to \exists \bar{y} \psi(\bar{x}, \bar{y})$$

- Premises are single relational atoms

Very used in practice!

Under LAV settings, canonical universal solutions are of polynomial size (combined complexity)

Lower combined complexity if we restrict to $\textsc{Lav}$ settings

Theorem

Input: $\textsc{Lav}$ setting $\mathcal{M}$, query Q, instance I and tuple $\bar{t}$
Problem: Is $\bar{t}$ in the certain answers of Q for I under $\mathcal{M}$?

$\textsc{Exptime}$-complete for $\textsc{Datalog}^{\mathsf{C}(\neq)}$ programs

Lower combined complexity if we restrict to $\textsc{Lav}$ settings

Theorem

Input: $\textsc{Lav}$ setting $\mathcal{M}$, query Q, instance I and tuple $\bar{t}$
Problem: Is $\bar{t}$ in the certain answers of Q for I under $\mathcal{M}$?

$$\begin{array}{rl} \textsc{Exptime}\text{-complete} & \text{for } \textsc{Datalog}^{\mathbf{C}(\neq)} \text{ programs} \\ \mathrm{NP}\text{-complete} & \text{for 1-}\mathrm{CQ}^{\neq} \end{array}$$

Lower combined complexity if we restrict to $\textsc{Lav}$ settings

Theorem

Input: $\textsc{Lav}$ setting $\mathcal{M}$, query Q, instance I and tuple $\bar{t}$
Problem: Is $\bar{t}$ in the certain answers of Q for I under $\mathcal{M}$?

$\textsc{Exptime}$-complete	for $\textsc{Datalog}^{\mathbf{C}(\neq)}$ programs
NP-complete	for 1-$\mathrm{CQ}^{\neq}$
Π_2^p-complete	for k-$\mathrm{CQ}^{\neq}$, $k \geq 2$
Π_2^p-complete	for $\mathrm{CQ}^{\neq}$

Lower combined complexity if we restrict to $\textsc{Lav}$ settings

Theorem

Input: $\textsc{Lav}$ setting $\mathcal{M}$, query Q, instance I and tuple $\bar{t}$
Problem: Is $\bar{t}$ in the certain answers of Q for I under $\mathcal{M}$?

$\textsc{Exptime}$-complete	for $\textsc{Datalog}^{\mathsf{C}(\neq)}$ programs
NP-complete	for 1-$\mathrm{CQ}^{\neq}$
Π_2^p-complete	for k-$\mathrm{CQ}^{\neq}$, $k \geq 2$
Π_2^p-complete	for $\mathrm{CQ}^{\neq}$

- Same results hold for unions

Outline

Formalization

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

Beyond union of conjunctive queries

- Expressive power of $\textsc{Datalog}^{\mathsf{C}(\neq)}$
- New tractable classes of queries

Combined Complexity

- $\textsc{Datalog}^{\mathsf{C}(\neq)}$ and queries with inequalities
- Restricting to $\textsc{Lav}$ settings

Concluding remarks

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

Study its properties

- Preserved under homomorphisms
- Certain answers can be computed in polynomial time (data complexity)

$\textsc{Datalog}^{\mathsf{C}(\neq)}$, a tractable language that express negation:

- Union of conjunctive queries with one negated atom per disjunct
- A fragment of 2-$\mathrm{UCQ}^{\neq}$

We can use $\textsc{Datalog}^{\mathsf{C}(\neq)}$ to find tractable classes of queries

Outline

Formalization

- $\text{DATALOG}^{\mathbf{C}(\neq)}$ programs

Beyond union of conjunctive queries

- Expressive power of $\text{DATALOG}^{\mathbf{C}(\neq)}$
- New tractable classes of queries

Combined Complexity

- $\text{DATALOG}^{\mathbf{C}(\neq)}$ and queries with inequalities
- Restricting to LAV settings

Concluding remarks