Finite Model Theory

Unit 4

Dan Suciu

Spring 2018
Unit 4: Query Containment and Equivalence
Resources

- Abitebou, Hull, Vianu, *Database Theory* (Alice book)

  https://simons.berkeley.edu/workshops/logic2016-boot-camp
  See Kolaiti’s tutorial on Logic and Databases

- Cerignou, Vollmer, *Boolean Constraint Satisfaction Problem*. 
Query

Fix a vocabulary $\sigma$.

An FO query is defined by formula $Q(x)$ with $k$ free variables $Q$ maps $A \in \text{STRUCT}[\sigma]$ to the relation $Q(A) \subseteq A^k$:

$$Q(A) \overset{\text{def}}{=} \{ a \subseteq A^k \mid A \models Q[a] \}$$

Discuss connection to FO reduction $\text{STRUCT}[\sigma] \rightarrow \text{STRUCT}[\tau]$.

When $k = 0$ then we call it a Boolean query: $Q(D)$ is true or false.

Warning: we use conflicting notations $Q(A)$ and $Q(x)$.
Problem Definition

**Definition (Query Containment)**

We say that $Q_1$ is contained in $Q_2$, $Q_1 \subseteq Q_2$ if for all $A$, $Q_1(A) \subseteq Q_2(A)$. The **containment problem** for a language $L$ is: given $Q_1, Q_2 \in L$ check if $Q_1 \subseteq Q_2$.

When $Q_1, Q_2$ are Boolean queries, then containment is logical implication: $Q_1 \rightarrow Q_2$.

**Definition (Query Equivalence)**

We say that $Q_1$ is equivalent to $Q_2$, $Q_1 \equiv Q_2$ if for all $A$, $Q_1(A) = Q_2(A)$. The **equivalence problem** for a language $L$ is: given $Q_1, Q_2 \in L$ check if $Q_1 \equiv Q_2$. 
**Problem Definition**

**Definition (Query Containment)**

We say that $Q_1$ is contained in $Q_2$, $Q_1 \subseteq Q_2$ if forall $A$, $Q_1(A) \subseteq Q_2(A)$. 

The **containment problem** for a language $L$ is:

given $Q_1, Q_2 \in L$ check if $Q_1 \subseteq Q_2$.

When $Q_1, Q_2$ are Boolean queries, then containment is logical implication: $Q_1 \rightarrow Q_2$.

**Definition (Query Equivalence)**

We say that $Q_1$ is equivalent to $Q_2$, $Q_1 \equiv Q_2$ if forall $A$, $Q_1(A) = Q_2(A)$.

The **equivalence problem** for a language $L$ is:

given $Q_1, Q_2 \in L$ check if $Q_1 \equiv Q_2$. 
Discussion

- If $L$ is closed under $\land$ or closed under $\lor$ then containment and equivalence have the same complexity. \textit{proof in class}

  Thus, containment and equivalence are essentially the same problem.

- However, it is undecidable for FO:

\begin{theorem}

The problem “given $Q_1, Q_2 \in FO$, is $Q_1 \subseteq Q_2$?” is undecidable.

\end{theorem}

proof in class

- Thus, we study containment for fragments $L \subseteq FO$. 
The Homomorphism Problem

Fix two structures $A = (A, R_1^A, \ldots, R_m^A)$, $B = (B, R_1^B, \ldots, R_m^B)$.

A homomorphism $f : A \rightarrow B$ is a function $f : A \rightarrow B$ s.t. $f(R_j^A) \subseteq R_j^B$ for $j = 1, m$.

**Definition (The Homomorphism Problem)**

The homomorphism problem is: given two structures $A, B$, check if there exists a homomorphism $h : A \rightarrow B$
The Homomorphism Problem: Complexity

Find $f : A \rightarrow B$

**Theorem**

(1) The homomorphism problem is NP-hard in general.
(2) There exists a fixed $B$ s.t. the homomorphism problem is NP-hard.

Prove (2) in class, twice: 3-colorability (ternary domain of $B$), 3SAT (binary domain of $B$).
Conjunctive Query

A Conjunctive Query (CQ) is a query of the form:

\[ Q(x) = \exists y (R_{j_1}(u_1) \land R_{j_1}(u_1) \land \ldots) \]

We often write it in datalog notation, dropping \( \exists \):

\[ Q(x) \leftarrow R_{j_1}(u_1) \land R_{j_1}(u_1) \land \ldots \]

Each \( R_{j_i}(u_i) \) is called an atom, or a subgoal.
Homomorphism and CQ Evaluation

The **canonical database** of a Boolean CQ $Q$, denoted $Q^D$, is the following:

- **Domain** = $\{x_1, \ldots, x_n\}$ (all variables of $Q$)
- **Relation** $R_j^{Q^D} = \text{all atoms } R_j(u)$ in $Q$.

E.g.: $Q = R(x, y) \land R(z, y) \land S(z, x)$

CQ evaluation is the same as the homomorphism problem:

**Fact**

For any structure (database) $D$, $D \models Q$ iff there exists a homomorphism $Q^D \to D$. 

Homomorphism and CQ Evaluation

The canonical database of a Boolean CQ $Q$, denoted $Q^D$, is the following:

- **Domain** = $\{x_1, \ldots, x_n\}$ (all variables of $Q$)
- **Relation** $R_j^Q = \text{all atoms } R_j(u) \text{ in } Q$.

E.g.: $Q = R(x, y) \land R(z, y) \land S(z, x)$

CQ evaluation is the same as the homomorphism problem:

**Fact**

For any structure (database) $D$, $D \models Q$ iff there exists a homomorphism $Q^D \rightarrow D$. 
Homomorphism and CQ Evaluation

The canonical database of a Boolean CQ $Q$, denoted $Q^D$, is the following:

- Domain = $\{x_1, \ldots, x_n\}$ (all variables of $Q$)
- Relation $R^Q_j = \text{all atoms } R_j(u)$ in $Q$.

E.g.: $Q = R(x, y) \land R(z, y) \land S(z, x)$

CQ evaluation is the same as the homomorphism problem:

<table>
<thead>
<tr>
<th>$Q^D$</th>
<th>$R$</th>
<th>$S$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$x$</td>
<td>$y$</td>
</tr>
<tr>
<td></td>
<td>$z$</td>
<td>$y$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fact

For any structure (database) $D$, $D \models Q$ iff there exists a homomorphism $Q^D \rightarrow D$. 
The Constraint Satisfaction Problem (CSP)

Fix a domain $D$ and a set of logical relations, $D = (R_1^D, \ldots, R_m^D)$.
Fix $n$ variables $x_1, \ldots, x_n$.
A constraint is an expression $R_j(x_{i_1}, \ldots, x_{i_k})$.

Definition

A Constraint Satisfaction Problem is a set $Q$ of constraints.
A solution is $f : \{x_1, \ldots, x_n\} \to D$ s.t. for every constraint $R_j(x_{i_1}, \ldots, x_{i_k})$, $(f(x_{j_1}), \ldots, f(x_{j_k})) \subseteq R_j^D$.

If $D = \{0, 1\}$ then we call it a Boolean CSP.
The Constraint Satisfaction Problem (CSP)

Fix a domain $D$ and a set of logical relations, $D = (R_1^D, \ldots, R_m^D)$. Fix $n$ variables $x_1, \ldots, x_n$. A constraint is an expression $R_j(x_{i_1}, \ldots, x_{i_k})$.

**Definition**

A **Constraint Satisfaction Problem** is a set $Q$ of constraints. A solution is $f : \{x_1, \ldots, x_n\} \rightarrow D$ s.t. for every constraint $R_j(x_{i_1}, \ldots, x_{i_k})$, $(f(x_{j_1}), \ldots, f(x_{j_k})) \subseteq R_j^D$.

If $D = \{0, 1\}$ then we call it a Boolean CSP.
Examples

3-colorability. $Q = \text{the graph};$ logical relation =

$$E^D: \begin{array}{cc}
\text{red} & \text{green} \\
\text{red} & \text{blue} \\
\text{green} & \text{blue}
\end{array}$$

3SAT is a CSP in class
Examples

3-colorability. \( Q = \) the graph; logical relation = 

<table>
<thead>
<tr>
<th>( E^D )</th>
</tr>
</thead>
</table>
| red      | green    
| red      | blue     
| green    | blue     

3SAT is a CSP in class
Homomorphism and the CSP

**Fact**

The CSP problem has a solution iff there exists a homomorphism \( Q \rightarrow D \).

The homomorphism goes from the problem \( Q \) to the logical relations \( D \).
Discussion

- CQ Evaluation and CSP are the same thing! And they are the same as the homomorphism problem:

\[ f : A \rightarrow B \]

- But they look at different “sides”:
  - CSP: fix logical relations \( B \), the input is the problem \( A \). NP-hard in general. Schaefer’s dichotomy for Boolean CSP into PTIME v.s. NP-hard.
  - CQ: fix the query \( A \), the input is the database \( B \). Always in PTIME (data complexity).
The Homomorphism Theorem for Containment of CQ

Consider Boolean queries only; extension to non-Boolean is straightforward.

**Theorem**

Let $Q_1, Q_2$ be CQ. The following are equivalent:

- $Q_1 \subseteq Q_2$
- There exists a homomorphism $f : Q_2 \rightarrow Q_1$.
- $Q_2$ is true on the canonical database given by $Q_1$.

**Consequence:** $Q_1 \equiv Q_2$ iff there exists two homomorphisms $Q_2 \rightarrow Q_1$ and $Q_1 \rightarrow Q_2$. 
Example

In class prove that \( Q_3 \subseteq Q_2 \equiv Q_1 \):

\[
Q_1 \leftarrow E(x, y), E(z, y), E(z, u), E(u, v)
\]
\[
Q_2 \leftarrow E(r, s), E(s, t)
\]
\[
Q_3 \leftarrow E(a, b), E(b, c), E(c, d)
\]
CQ Query Minimization

A CQ $Q$ is called *minimal* if:
for all $Q'$, if $Q' \equiv Q$, then $Q'$ has at least as many atoms as $Q$.

Theorem

If $Q \equiv Q'$ and both are minimal, then $Q$, $Q'$ are isomorphic.

Proof. Let $f : Q \rightarrow Q'$, $g : Q' \rightarrow Q$ be two homomorphisms.

Then $g \circ f : Q \rightarrow Q$ is also a homomorphism.

Since $Q$ is minimal, $g \circ f$ must be surjective. why?

Since the body of $Q$ is finite (has finitely many atoms), $g \circ f$ is a bijection.

Hence both $f, g$ are bijections, i.e. isomorphisms.
CQ Query Minimization

A CQ $Q$ is called *minimal* if:
forall $Q'$, if $Q' \equiv Q$, then $Q'$ has at least as many atoms as $Q$.

**Theorem**

*If $Q \equiv Q'$ and both are minimal, then $Q$, $Q'$ are isomorphic.*

Proof. Let $f : Q \rightarrow Q'$, $g : Q' \rightarrow Q$ be two homomorphisms.

Then $g \circ f : Q \rightarrow Q$ is also a homomorphism.

Since $Q$ is minimal, $g \circ f$ must be surjective. why?

Since the body of $Q$ is finite (has finitely many atoms), $g \circ f$ is a bijection.

Hence both $f, g$ are bijections, i.e. isomorphisms.
CQ Query Minimization

A CQ $Q$ is called *minimal* if:
for all $Q'$, if $Q' \equiv Q$, then $Q'$ has at least as many atoms as $Q$.

**Theorem**

*If $Q \equiv Q'$ and both are minimal, then $Q$, $Q'$ are isomorphic.*

**Proof.** Let $f : Q \to Q'$, $g : Q' \to Q$ be two homomorphisms.

Then $g \circ f : Q \to Q$ is also a homomorphism.

Since $Q$ is minimal, $g \circ f$ must be surjective. *why?*

Since the body of $Q$ is finite (has finitely many atoms), $g \circ f$ is a bijection.

Hence both $f, g$ are bijections, i.e. isomorphisms.
CQ Query Minimization

A CQ $Q$ is called *minimal* if:
for all $Q'$, if $Q' \equiv Q$, then $Q'$ has at least as many atoms as $Q$.

**Theorem**

*If $Q \equiv Q'$ and both are minimal, then $Q$, $Q'$ are isomorphic.*

Proof. Let $f : Q \to Q'$, $g : Q' \to Q$ be two homomorphisms.

Then $g \circ f : Q \to Q$ is also a homomorphism.

Since $Q$ is minimal, $g \circ f$ must be surjective. why?

Since the body of $Q$ is finite (has finitely many atoms), $g \circ f$ is a bijection.

Hence both $f, g$ are bijections, i.e. isomorphisms.
CQ Query Minimization

A CQ $Q$ is called *minimal* if:
forall $Q'$, if $Q' \equiv Q$, then $Q'$ has at least as many atoms as $Q$.

**Theorem**

*If $Q \equiv Q'$ and both are minimal, then $Q$, $Q'$ are isomorphic.*

Proof. Let $f : Q \rightarrow Q'$, $g : Q' \rightarrow Q$ be two homomorphisms.

Then $g \circ f : Q \rightarrow Q$ is also a homomorphism.

Since $Q$ is minimal, $g \circ f$ must be surjective. *why?*

Since the body of $Q$ is finite (has finitely many atoms), $g \circ f$ is a bijection.

Hence both $f, g$ are bijections, i.e. isomorphisms.
CQ Query Minimization

A CQ \( Q \) is called *minimal* if:
forall \( Q' \), if \( Q' \equiv Q \), then \( Q' \) has at least as many atoms as \( Q \).

**Theorem**

*If \( Q \equiv Q' \) and both are minimal, then \( Q, Q' \) are isomorphic.*

**Proof.** Let \( f : Q \rightarrow Q' \), \( g : Q' \rightarrow Q \) be two homomorphisms.

Then \( g \circ f : Q \rightarrow Q \) is also a homomorphism.

Since \( Q \) is minimal, \( g \circ f \) must be surjective. *why?*

Since the body of \( Q \) is finite (has finitely many atoms), \( g \circ f \) is a bijection.

Hence both \( f, g \) are bijections, i.e. isomorphisms.
The Minimization Procedure

Given $Q$, we want to find the (unique) minimal query $Q_m$ s.t. $Q \equiv Q_m$.

1. Start with $Q' = Q$.

2. For each atom $R_j$ of $Q'$, check if there exists a homomorphism $f : Q' \rightarrow Q' \setminus \{R_j\}$; if yes, then set $Q' = Q' \setminus \{R_j\}$ and continue.

3. If no such $R_j$ exits, then stop and return $Q_m = Q'$.

Prove in class: this procedure returns the unique minimal query equivalent to $Q$.

Note: the minimal query is always a subset of the atoms of $Q$!
The Minimization Procedure

Given $Q$, we want to find the (unique) minimal query $Q_m$ s.t. $Q \equiv Q_m$.

(1) Start with $Q' = Q$.

(2) For each atom $R_j$ of $Q'$, check if there exists a homomorphism $f : Q' \to Q' - \{R_j\}$; if yes, then set $Q' = Q' - \{R_j\}$ and continue.

(3) If no such $R_j$ exits, then stop and return $Q_m = Q'$.

Prove in class: this procedure returns the unique minimal query equivalent to $Q$.

Note: the minimal query is always a subset of the atoms of $Q$!
The Minimization Procedure

Given $Q$, we want to find the (unique) minimal query $Q_m$ s.t. $Q \equiv Q_m$.

(1) Start with $Q' = Q$.

(2) For each atom $R_j$ of $Q'$, check if there exists a homomorphism $f : Q' \to Q' - \{R_j\}$; if yes, then set $Q' = Q' - \{R_j\}$ and continue.

(3) If no such $R_j$ exits, then stop and return $Q_m = Q'$.

Prove in class: this procedure returns the unique minimal query equivalent to $Q$.

Note: the minimal query is always a subset of the atoms of $Q$!
The Minimization Procedure

Given $Q$, we want to find the (unique) minimal query $Q_m$ s.t. $Q \equiv Q_m$.

(1) Start with $Q' = Q$.

(2) For each atom $R_j$ of $Q'$, check if there exists a homomorphism $f : Q' \rightarrow Q' - \{R_j\}$; if yes, then set $Q' = Q' - \{R_j\}$ and continue.

(3) If no such $R_j$ exits, then stop and return $Q_m = Q'$.

Prove in class: this procedure returns the unique minimal query equivalent to $Q$.

Note: the minimal query is always a subset of the atoms of $Q$!
The Minimization Procedure

Given $Q$, we want to find the (unique) minimal query $Q_m$ s.t. $Q \equiv Q_m$.

(1) Start with $Q' = Q$.

(2) For each atom $R_j$ of $Q'$, check if there exists a homomorphism $f : Q' \rightarrow Q' - \{R_j\}$; if yes, then set $Q' = Q' - \{R_j\}$ and continue.

(3) If no such $R_j$ exits, then stop and return $Q_m = Q'$.

Prove in class: this procedure returns the unique minimal query equivalent to $Q$.

Note: the minimal query is always a subset of the atoms of $Q$!
The Minimization Procedure

Given $Q$, we want to find the (unique) minimal query $Q_m$ s.t. $Q \equiv Q_m$.

(1) Start with $Q' = Q$.

(2) For each atom $R_j$ of $Q'$, check if there exists a homomorphism $f : Q' \rightarrow Q' - \{R_j\}$; if yes, then set $Q' = Q' - \{R_j\}$ and continue.

(3) If no such $R_j$ exists, then stop and return $Q_m = Q'$.

Prove in class: this procedure returns the unique minimal query equivalent to $Q$.

Note: the minimal query is always a subset of the atoms of $Q$!
Discussion

- CQ query evaluation is CSP \textit{from the other side}, and in PTIME.

- CQ query containment/equivalence is CSP \textit{from both ends}, and NP-complete.

- To minimize $Q$, simply remove atoms one by one, in any order, until no other removal is possible.

- If $G$ is a graph, then a core is a subgraph $G_0 \subseteq G$ s.t. (a) there exists a homomorphism $G \rightarrow G_0$, and (b) $G_0$ is smallest with this property. Is the core unique? How does one find it?
Clauses

A Knowledge Base (in AI) is often described by a collection of *clauses*:

\[ C = \forall x (L_1 \lor L_2 \lor \ldots) \]

where each literal is some \( R(u) \) or \( \neg R(u) \).

**Fact**

*If \( C, C' \) are two positive clauses (w/o negation) then the implication problem \( C \rightarrow C' \) is decidable and co-NP complete.*

*proof in class (reduction to CQ)*

Note: this fact seems little known!
Unions of Conjunctive Queries

A Conjunctive Query (CQ) is a query of the form:

$$Q(x) = \exists y (R_{j_1}(u_1) \land R_{j_1}(u_1) \land \ldots)$$

A Union of Conjunctive Queries (UCQ) is a query of the form:

$$Q(x) = Q_1(x) \lor Q_2(x) \lor \ldots$$

where $Q_1, Q_2, \ldots$ are CQ’s with the same free variables.
Example

Equivalently, a UCQ is a non-recursive datalog program. Example:

\[
\begin{align*}
P_1(x, y) & \leftarrow E(x, y) \\
P_2(x, y) & \leftarrow P_1(x, y) \\
P_3(x, y) & \leftarrow P_2(x, y) \\
P_4(x, y) & \leftarrow P_3(x, y) \\
Q(x, y) & \leftarrow P_4(x, y)
\end{align*}
\]

\[
\begin{align*}
P_2(x, y) & \leftarrow P_1(x, z) \land P_1(z, y) \\
P_3(x, y) & \leftarrow P_2(x, z) \land P_2(z, y) \\
P_4(x, y) & \leftarrow P_3(x, z) \land P_3(z, y) \\
Q & \leftarrow P_4(x, z) \land P_4(z, y)
\end{align*}
\]

How much larger is the UCQ compared to the datalog program?
Containment for UCQ

We discuss Boolean queries only; non-Boolean queries are handled similarly, straightforwardly:

\[ Q = Q_1 \lor Q_2 \lor \ldots \lor Q_m \]
\[ Q' = Q'_1 \lor Q'_2 \lor \ldots \lor Q'_n \]

Theorem

\( Q \subseteq Q' \) iff \( \forall i \exists j \text{ such that } Q_i \subseteq Q'_j \). Hence, containment of UCQ is \( NP \)-complete.

Proof in class
Minimizing UCQ

\[ Q = Q_1 \lor Q_2 \lor \ldots \lor Q_m \]

(1) Minimize each CQ \( Q_j \).

(2) For all \( i \), if there exists \( j \) s.t. \( Q_i \subseteq Q_j \), then remove \( Q_i \).

(3) The remaining query is minimal, and unique up to isomorphism. \text{proof in class}
Domain-Independent Queries

$Q$ is called *domain-independent* if for any two structures $D, D'$ with the same relations but different domains, we have $Q(D) = Q(D')$:

$D = (D, R_1^D, \ldots, R_m^D)$

$D' = (D', R_1^D, \ldots, R_m^D)$

Which queries are domain independent?

$\exists x \exists y R(x, y)$

$\exists x \exists y (R(x) \land \neg S(x, y))$

$\forall y S(y)$

$\exists x \exists y \neg R(x, y)$

$\exists x \exists y (R(x) \land \neg S(x, y) \land T(y))$

$\forall x \forall y (R(x, y) \rightarrow S(y))$

In databases we consider only domain-independent queries.

Checking if $Q$ is domain independent is undecidable in general *why?*
Monotone Queries

Two structures are contained, \( A \subseteq B \), if the domains and all their relations are contained: \( A \subseteq B, R^A_j \subseteq R^B_j, j = 1, m \).

A query \( Q \) is monotone if \( A \subseteq B \) implies \( Q(A) \subseteq Q(B) \).

Checking if \( Q \) is monotone is undecidable in general why?
More Query Languages

The languages $CQ^<$, $CQ^\neg$, $CQ^{<,\neg}$ extend CQ with $<$ or $\neg$ respectively; similarly UCQ.

Examples to which language do they belong?

$$\exists y \exists z \text{Friend}(x, y) \land \text{Friend}(y, z) \land \text{Boss}(z)$$
$$\exists y \exists z \text{Friend}(x, y) \land \text{Friend}(y, z) \land \neg \text{Boss}(z)$$
$$\exists y \exists z \text{Friend}(x, y) \land \text{Friend}(y, z) \land \text{Boss}(z) \land x < z$$

**In class** do we need $=$ in CQ, i.e. $CQ^=$?
## Summary of Query Languages

<table>
<thead>
<tr>
<th>Syntax</th>
<th>FO fragment</th>
<th>Domain independent?</th>
<th>Monotone?</th>
</tr>
</thead>
<tbody>
<tr>
<td>CQ</td>
<td>(FO(\exists, \land))</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>CQ(&lt;)</td>
<td>(FO(\exists, \land, &lt;))</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>CQ(\neg)</td>
<td>(FO(\exists, \land, \neg)) (Negation Normal Form)</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>UCQ</td>
<td>(FO(\exists, \lor, \land))</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>UCQ(&lt;)</td>
<td>(FO(\exists, \lor, \land, &lt;))</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>UCQ(\neg)</td>
<td>(FO(\exists, \lor, \land, \neg)) (Negation Normal Form)</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>
**Decidability**

**Theorem**

*The containment problem for UCQ\(\leq\), \(\neg\) is decidable.*

Proof: consider Boolean queries only. Any UCQ\(\leq\), \(\neg\) query can be written as \(\exists x \varphi(x)\). Then:

\[
\begin{align*}
Q_1 \subseteq Q_2 & \text{ iff } \models \exists x \varphi_1(x) \rightarrow \exists y \varphi_2(y) \\
& \text{ iff } \models (\neg \exists x \varphi_1(x)) \lor (\exists y \varphi_2(y)) \\
& \text{ iff } \models (\forall x \neg \varphi_1(x)) \lor (\exists y \varphi_2(y)) \\
& \text{ iff } \models \forall x \exists y (\neg \varphi_1(x) \lor \varphi_2(y))
\end{align*}
\]

The latter is the negation of a Bernays-Schönfinkel formula \(\exists^* \forall^*\), hence validity is decidable.
Containment Procedure for $CQ^<$

Main idea: it is insufficient to treat $<$ as any other predicate.

$$Q_1 = R(x, y) \land R(y, z) \land x < z$$

$$Q_2 = R(u, v), \quad u < v$$

Then $Q_1 \subseteq Q_2$ why? yet there is no homomorphism $Q_2 \rightarrow Q_1$ that maps $u < v$ to some $<$-atom.

Solution: expand $Q_1$ by considering all linear orders of variables

$$Q_{11} = R(x, y) \land R(y, z) \land y < x < z$$

$$Q_{12} = R(x, y) \land R(y, z) \land x = y < z$$

$$Q_{13} = R(x, y) \land R(y, z) \land x < z < y$$

$$Q_{14} = R(x, y) \land R(y, z) \land x < y = z$$

$$Q_{15} = R(x, y) \land R(y, z) \land x < z \land x < y$$

Prove in class: $Q_1 \equiv Q_{11} \lor \cdots \lor Q_{15} \subseteq Q_2$.

**Theorem**

The Containment problem for $CQ^<$ (and for $UCQ^<$) is $\Pi^p_2$-complete.
Negation

Once we add negation, a query may not be domain independent. Problem: the abbreviated syntax suggests two interpretations. E.g.

\[ Q \leftarrow R(x, y) \land \neg S(y, z) \]

Interpretation 1: \( \exists x \exists y \exists z (R(x, y) \land \neg S(y, z)) \)

Interpretation 2: the result of this datalog program:

\[ \text{Not}S(y) \leftarrow S(y, z) \]
\[ Q \leftarrow R(x, y) \land \text{Not}S(y) \]

Note: this means \( \exists x \exists y \forall z (R(x, y) \land \neg S(y, z)) \)
Negation: Interpretation 1

Theorem

\textit{Containment of CQ}^\neg \textit{ queries under interpretation 1 is } \Pi^p_2 \textit{ complete.}

Curiously, I could never find a published proof!
Negation: Interpretation 2

Theorem

Containment of $CQ^-$ queries under interpretation 2 is undecidable.
Discussion

- Containment of $\mathit{FO}(\exists, \lor, \land)$ is decidable (and in $\Pi_2^p$) because of Bernays-Schönfinkel.

- Better complexities (meaning NP) for various fragments.

- Checking containment $Q_1 \subseteq Q_2$ is related to query evaluation of $Q_2$ on some database(s) derived from $Q_1$.

- All results discussed here carry over to implication of universally quantified clauses. \textit{seems little known in the AI community}