CSE 544: Principles of Database Systems

Conjunctive Queries (and beyond)
Outline

• Conjunctive queries

• Query containment and equivalence

• Query minimization

• Undecidability for relational queries
Conjunctive Queries (CQ)

- CQ = one datalog rule
- CQ = SELECT-DISTINCT-FROM-WHERE
- CQ = select/project/join (σ, Π, ⊙) fragment of RA
- CQ = existential/conjunctive (∃, ∧) fragment of RC

Notice: strictly speaking we are not allowed to use <, ≤, >, ≥, ≠ in CQ’s. If we include these, then the language is called CQ<, or CQ≠, etc.
Examples

Find all employees with same manager as “Smith”:

q(x) :- ManagedBy(“Smith”,y), ManagedBy(x,y)

SELECT DISTINCT m2.name
FROM ManagedBy m1,
    ManagedBy m2
WHERE m1.name=“Smith”
    AND m1.manager=m2.manager
Examples

• Example of CQ

\[ q(x,y) = \exists z. (R(x,z) \land \exists u. (R(z,u) \land R(u,y))) \]
\[ q(x) = \exists z. \exists u. (R(x,z) \land R(z,u) \land R(u,y)) \]

• Examples of non-CQ:

\[ q(x,y) = \forall z. (R(x,z) \rightarrow R(y,z)) \]
\[ q(x) = T(x) \lor \exists z. S(x,z) \]
Query Equivalence and Containment

Containment/equivalence are examples of **static analysis**

Used in:

- Query optimization
- Query rewriting using views
Definition. Queries $q_1$ and $q_2$ are equivalent if for every database $D$, $q_1(D) = q_2(D)$.

Notation: $q_1 \equiv q_2$
Query Containment

**Definition.** Query $q_1$ is contained in $q_2$ if for every database $D$, $q_1(D) \subseteq q_2(D)$.

Notation: $q_1 \subseteq q_2$

**Fact:** $q_1 \subseteq q_2$ and $q_2 \subseteq q_1$ iff $q_1 \equiv q_2$

We discuss only query containment.
Example 1

Is $q_1 \subseteq q_2$?

$q_1(x) :- R(x,u), R(u,v), R(v,w)$
$q_2(x) :- R(x,u), R(u,v)$
Example 2

Is $q_1 \subseteq q_2$?

$q_1(x) :- R(x,u), R(u,v), R(v,x)$
$q_2(x) :- R(x,u), R(u,x)$
Example 3

Is $q_1 \subseteq q_2$?

$q_1(x,y) :- R(x,y)$
$q_2(x,y) :- R(x,u), R(v,u), R(v,y)$
Example 4

Is $q_1 \subseteq q_2$?

$q_1(x) :- R(x,u), R(u,v)$
$q_2(x) :- R(x,u), R(x,y), R(u,v), R(u,w)$
Example 5

Is $q_1 \subseteq q_2$?

$q_1(x) \ :- \ R(x,u), \ R(u,u)$
$q_2(x) \ :- \ R(x,u), \ R(u,v), \ R(v,w)$
Example 6

Is $q_1 \subseteq q_2$?

$q_1(x) :- R(x,u), R(u, "Smith")$

$q_2(x) :- R(x,u), R(u,v)$
Discussion

• We discuss only query containment for **Boolean** queries
  – If $q_1, q_2$ are **Boolean** queries, then containment $q_1(D) \subseteq q_2(D)$ means implication $q_1(D) \rightarrow q_2(D)$

• If $q_1, q_2$ are **not** **Boolean**, then convert them into **Boolean** queries by pretending that their head variables are constants
  – We must make *the same* constants in both $q_1, q_2$
Theorem [Chandra&Harel’1977]
Given two CQ, $q_1$, $q_2$, the following are equivalent:
1. For every database $D$, $q_1(D) \subseteq q_2(D)$
2. There exists a homomorphism $h : q_2 \rightarrow q_1$
3. $q_2$ is true on the canonical database of $q_1$
Moreover, this problem is NP-complete
Query Homomorphisms

Definition A homomorphism $h : q_2 \rightarrow q_1$ is a function $h : \text{var}(q_2) \rightarrow \text{var}(q_1) \cup \text{const}(q_1)$ such that, for every atom $R(x, y, z, \ldots)$ in the query $q_2$, there is an atom $R(h(x), h(y), h(z), \ldots)$ in the query $q_1$.

The Homomorphism Theorem $q_1 \subseteq q_2$ iff there exists a homomorphism $h : q_2 \rightarrow q_1$. 
Example 1

Is \( q_1 \subseteq q_2 \)?

\[
q_1(x) :\quad R(x,u), \ R(u,v), \ R(v,w)
\]

\[
q_2(x) :\quad R(x,u), \ R(u,v)
\]
Example 1

Is \( q_1 \subseteq q_2 \) ?

\[
q_1(x) :- R(x,u), R(u,v), R(v,w)
\]

\[
q_2(x) :- R(x,u), R(u,v)
\]

First, transform the head variables into a constant: must be the same in \( q_1 \) and \( q_2 \)!

\[
q_1 :- R("x",u), R(u,v), R(v,w)
\]

\[
q_2 :- R("x",u), R(u,v)
\]
Example 1

Is \( q_1 \subseteq q_2 \) ?

\[
q_1(x) :- R(x,u), R(u,v), R(v,w) \\
q_2(x) :- R(x,u), R(u,v)
\]

First, transform the head variables into a constant: must be the same in \( q_1 \) and \( q_2 \)!

\[
q_1 :- R("x",u), R(u,v), R(v,w) \\
q_2 :- R("x",u), R(u,v)
\]

The homomorphism \( h : q_2 \to q_1 \) is the following:

\( h(u) = u, \ h(v) = v \)

In class: check that \( h \) is a homomorphism
Example 1

Is $q_1 \subseteq q_2$?

$q_1(x) : -$ R(x,u), R(u,v), R(v,w) 
$q_2(x) : -$ R(x,u), R(u,v)

First, transform the head variables into
a constant: must be the same in $q_1$ and $q_2$!

$q_1 : -$ R(“x”,u), R(u,v), R(v,w) 
$q_2 : -$ R(“x”,u), R(u,v)

The homomorphism $h : q_2 \rightarrow q_1$ is the following:
$h(u) = u, \ h(v) = v$

This proves $q_1 \subseteq q_2$

In class: check that $h$ is a homomorphism
Example 2

Is $q_1 \subseteq q_2$?

$q_1(x) :- R(x,u), R(u,v), R(v,x)$
$q_2(x) :- R(x,u), R(u,x)$
Example 2

Is $q_1 \subseteq q_2$?

$q_1(x) :- R(x,u), R(u,v), R(v,x)$
$q_2(x) :- R(x,u), R(u,x)$

In class: check that there is no homomorphism

This proves $q_1 \not\subseteq q_2$
Example 3

Is $q_1 \subseteq q_2$ ?

$q_1(x,y) :- R(x,y)$
$q_2(x,y) :- R(x,u), R(v,u), R(v,y)$
Example 3

Is $q_1 \subseteq q_2$?

$q_1(x,y) :- \text{R}(x,y)$
$q_2(x,y) :- \text{R}(x,u), \text{R}(v,u), \text{R}(v,y)$

Head variables $x,y$ are treated like constants

The homomorphism $h : q_2 \rightarrow q_1$ is the following:
$h(u) = y$, $h(v) = x$

This proves $q_1 \subseteq q_2$
Canonical Database

**Definition.** Given a conjunctive query $q$, the canonical database $D_q$ is the following:

- The active domain $= \text{vars}(q) \cup \text{const}(q)$
- Each atom in $q$ becomes a tuple in $D_q$

**The Canonical Database Theorem** $q_1 \subseteq q_2$ iff the query $q_2$ is true on the canonical database $D_{q_1}$.
Example 1

Is $q_1 \subseteq q_2$?

$q_1(x) :- R(x,u), R(u,v), R(v,w)$
$q_2(x) :- R(x,u), R(u,v)$
Example 1

Is $q_1 \subseteq q_2$?

$q_1(x) :- R(x,u), R(u,v), R(v,w)$
$q_2(x) :- R(x,u), R(u,v)$

$D_{q_1}$

<table>
<thead>
<tr>
<th></th>
<th>x</th>
<th>u</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>v</td>
<td>w</td>
</tr>
</tbody>
</table>

$R:$

Compute $q_2$ on $D_{q_1}$: the answer is true

Keep in mind that “x” in $q_2$ is a constant, it can only match x.

This proves $q_1 \subseteq q_2$
Example 2

Is $q_1 \subseteq q_2$?

$q_1(x) :- R(x,u), R(u,v), R(v,x)$
$q_2(x) :- R(x,u), R(u,x)$
Example 2

Is $q_1 \subseteq q_2$?

$q_1(x) :- R(x,u), R(u,v), R(v,x)$
$q_2(x) :- R(x,u), R(u,x)$

\[ D_{q_1} \]

<table>
<thead>
<tr>
<th>R:</th>
<th>x</th>
<th>u</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>u</td>
<td>v</td>
</tr>
<tr>
<td></td>
<td>v</td>
<td>x</td>
</tr>
</tbody>
</table>

Compute $q_2$ on $D_{q_1}$: the answer is \text{false}

This proves that $q_1 \not\subseteq q_2$
Query Containment for CQ

Theorem [Chandra&Harel’1977]
Given two CQ, $q_1$, $q_2$, the following are equivalent:
1. For every database $D$, $q_1(D) \subseteq q_2(D)$
2. There exists a homomorphism $h : q_2 \rightarrow q_1$
3. $q_2$ is true on the canonical database of $q_1$

Moreover, this problem is NP-complete

Proof: show in class that $1 \rightarrow 3 \rightarrow 2 \rightarrow 1$
The Complexity

**Theorem** Checking containment of two CQ queries is NP-complete
Reduction from 3SAT

Given a 3CNF $\Phi$

Step 1:
  construct $q_1$ independently of $\Phi$. 
Proof

Reduction from 3SAT

Given a 3CNF $\Phi$

**Step 1:**
- construct $q_1$ independently of $\Phi$.

**Step 2:**
- construct $q_2$ from $\Phi$. 
Proof

Reduction from 3SAT

Given a 3CNF $\Phi$

**Step 1:**
construct $q_1$ independently of $\Phi$.

**Step 2:**
construct $q_2$ from $\Phi$.

**Claim:**
there exists a homomorphism $q_2 \rightarrow q_1$
iff $\Phi$ is satisfiable
Proof

Reduction from 3SAT

Given a 3CNF $\Phi$

Step 1:
construct $q_1$ independently of $\Phi$.

Step 2:
construct $q_2$ from $\Phi$.

Claim:
there exists a homomorphism $q_2 \rightarrow q_1$
iff $\Phi$ is satisfiable

Running example:
$\Phi = (\neg X_3 \lor \neg X_1 \lor X_4) (X_1 \lor X_2 \lor X_3) (\neg X_2 \lor \neg X_3 \lor X_1)$
Step 1: Constructing $q_1$

There are four types of clauses possible in a 3SAT:

Type 1 = $\neg X \lor \neg Y \lor \neg Z$
Step 1: Constructing $q_1$

There are four types of clauses possible in a 3SAT:

Type 1 = $\neg X \lor \neg Y \lor \neg Z$

Type 2 = $\neg X \lor \neg Y \lor Z$
Step 1: Constructing $q_1$

There are four types of clauses possible in a 3SAT:

Type 1  = $\neg X \lor \neg Y \lor \neg Z$

Type 2  = $\neg X \lor \neg Y \lor Z$

Type 3  = $\neg X \lor Y \lor Z$
Step 1: Constructing $q_1$

There are four types of clauses possible in a 3SAT:

- Type 1 = \neg X \lor \neg Y \lor \neg Z
- Type 2 = \neg X \lor \neg Y \lor Z
- Type 3 = \neg X \lor Y \lor Z
- Type 4 = X \lor Y \lor Z

For each type, we include in $q_1$ a relation with all 7 satisfying assignments $R$ (misses 1,1,1).
Step 1: Constructing $q_1$

There are four types of clauses possible in a 3SAT:

Type 1: $\neg X \lor \neg Y \lor \neg Z$
Type 2: $\neg X \lor \neg Y \lor Z$
Type 3: $\neg X \lor Y \lor Z$
Type 4: $X \lor Y \lor Z$

For each type, we include in $q_1$ a relation with all 7 satisfying assignments:

<table>
<thead>
<tr>
<th>$R$</th>
<th>$S$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(misses 1,1,1)</td>
<td>(misses 1,1,0)</td>
</tr>
<tr>
<td>0 0 0</td>
<td>0 0 0</td>
</tr>
<tr>
<td>0 0 1</td>
<td>0 0 1</td>
</tr>
<tr>
<td>0 1 0</td>
<td>0 1 0</td>
</tr>
<tr>
<td>0 1 1</td>
<td>0 1 1</td>
</tr>
<tr>
<td>1 0 0</td>
<td>1 0 0</td>
</tr>
<tr>
<td>1 0 1</td>
<td>1 0 1</td>
</tr>
<tr>
<td>1 1 0</td>
<td>1 1 1</td>
</tr>
</tbody>
</table>
Step 1: Constructing $q_1$

There are four types of clauses possible in a 3SAT:

- **Type 1** = $\neg X \lor \neg Y \lor \neg Z$
- **Type 2** = $\neg X \lor \neg Y \lor Z$
- **Type 3** = $\neg X \lor Y \lor Z$
- **Type 4** = $X \lor Y \lor Z$

For each type, we include in $q_1$ a relation with all 7 satisfying assignments:

<table>
<thead>
<tr>
<th>R (misses 1,1,1)</th>
<th>S (misses 1,1,0)</th>
<th>T (misses 1,0,0)</th>
<th>K (misses 0,0,0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 1</td>
</tr>
<tr>
<td>0 0 1</td>
<td>0 0 1</td>
<td>0 0 1</td>
<td>0 1 0</td>
</tr>
<tr>
<td>0 1 0</td>
<td>0 1 0</td>
<td>0 1 0</td>
<td>0 1 1</td>
</tr>
<tr>
<td>0 1 1</td>
<td>0 1 1</td>
<td>0 1 1</td>
<td>1 0 0</td>
</tr>
<tr>
<td>1 0 0</td>
<td>1 0 0</td>
<td>1 0 1</td>
<td>1 0 1</td>
</tr>
<tr>
<td>1 0 1</td>
<td>1 0 1</td>
<td>1 1 0</td>
<td>1 1 0</td>
</tr>
<tr>
<td>1 1 0</td>
<td>1 1 1</td>
<td>1 1 1</td>
<td>1 1 1</td>
</tr>
</tbody>
</table>
Step 1: Constructing $q_1$

There are four types of clauses possible in a 3SAT:

- **Type 1** = $\neg X \lor \neg Y \lor \neg Z$
- **Type 2** = $\neg X \lor \neg Y \lor Z$
- **Type 3** = $\neg X \lor Y \lor Z$
- **Type 4** = $X \lor Y \lor Z$

For each type, we include in $q_1$ a relation with all 7 satisfying assignments.

R (misses 1,1,1)

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

S (misses 1,1,0)

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

T (misses 1,0,0)

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

K (misses 0,0,0)

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

So which are the atoms in $q_1$?
Step 2: Constructing $q_2$

Given $\Phi$, we construct $q_2$ as follows.

For each clause in $\Phi$ we create an atom in $q_2$:

- If the clause is of Type 1, then the atom is $R$;
  if the clause is of Type 2, then the atom is $S$; etc.
- The variables in this atom are the same as the variables in the clause
Step 2: Constructing $q_2$

Given $\Phi$, we construct $q_2$ as follows.

For each clause in $\Phi$ we create an atom in $q_2$:
- If the clause is of Type 1, then the atom is $R$;
  - if the clause is of Type 2, then the atom is $S$; etc.
- The variables in this atom are the same as the variables in the clause

Example:

$\Phi = (\neg X_3 \lor \neg X_1 \lor X_4) \land (X_1 \lor X_2 \lor X_3) \land (\neg X_2 \lor \neg X_3 \lor X_1)$
Step 2: Constructing $q_2$

Given $\Phi$, we construct $q_2$ as follows.

For each clause in $\Phi$ we create an atom in $q_2$:

- If the clause is of Type 1, then the atom is $R$; if the clause is of Type 2, then the atom is $S$; etc.
- The variables in this atom are the same as the variables in the clause

Example:

$\Phi = (\neg X_3 \lor \neg X_1 \lor X_4) \land (X_1 \lor X_2 \lor X_3) \land (\neg X_2 \lor \neg X_3 \lor X_1)$

$q_2 = S(x_3,x_1,x_4), K(x_1,x_2,x_3), S(x_2,x_3,x_1)$
Proof

Claim: There exists a homomorphism $q_2 \rightarrow q_1$ iff $\Phi$ is satisfiable

1. Suppose there is a satisfying assignment $\theta$ for $\Phi$:
   - Define the function $h: \text{Vars}(q_2) \rightarrow \text{Const}(q_1)$:
     • If $\theta(X_i) = 0$ then $h(x_i) = 0$
     • If $\theta(X_i) = 1$ then $h(x_i) = 1$
   - $h$ is a homomorphism $h : q_2 \rightarrow q_1$ (why??)

2. Suppose there exists a homomorphism $h : q_2 \rightarrow q_1$.
   - Define the following assignment $\theta$:
     • If $h(x_i) = 0$ then $\theta(X_i) = 0$
     • If $h(x_i) = 1$ then $\theta(X_i) = 1$
   - $\theta$ is a satisfying assignment for $\Phi$ (why??)
Proof

Claim: There exists a homomorphism $q_2 \rightarrow q_1$ iff $\Phi$ is satisfiable

1. Suppose there is a satisfying assignment $\theta$ for $\Phi$:
   – Define the function $h$: $\text{Vars}(q_2) \rightarrow \text{Const}(q_1)$:
     • If $\theta(X_i) = 0$ then $h(x_i) = 0$
     • If $\theta(X_i) = 1$ then $h(x_i) = 1$
   – $h$ is a homomorphism $h : q_2 \rightarrow q_1$ (why??)

2. Suppose there exists a homomorphism $h : q_2 \rightarrow q_1$.
   – Define the following assignment $\theta$:
     • If $h(x_i) = 0$ then $\theta(X_i) = 0$
     • If $h(x_i) = 1$ then $\theta(X_i) = 1$
   – $\theta$ is a satisfying assignment for $\Phi$ (why??)

This completes the proof: query containment is NP-hard.
Discussion

• We have a procedure to check if two queries are equivalent
• Will use to do query optimization (discussed next)
• NP-hard is not that scary today: SAT solvers can scale to tens of thousands of clauses
Definition: A conjunctive query $q$ is minimal if for every other conjunctive query $q'$, if $q \equiv q'$ then $q'$ has at least as many atoms as $q$. 
Query Minimization

**Definition** A conjunctive query $q$ is minimal if for every other conjunctive query $q'$, if $q \equiv q'$ then $q'$ has at least as many atoms as $q$.

Are these queries minimal?

$q(x) :- R(x,y), R(y,z), R(x,v)$
**Query Minimization**

**Definition** A conjunctive query $q$ is minimal if for every other conjunctive query $q'$, if $q \equiv q'$ then $q'$ has at least as many atoms as $q$.

Are these queries minimal?

$q(x) :- R(x,y), R(y,z), R(x,v)$

$q(x) :- R(x,y), R(y,z), R(x, ’Alice’)$
Query Minimization

• Query minimization algorithm

  Choose an atom of $q$, and remove it: let $q'$ be the new query

  We already know $q \subseteq q'$ (why?)

  If $q' \subseteq q$ then permanently remove it; otherwise try another atom

• Notice: the order in which we inspect atoms doesn’t matter
Query Minimization In Practice

• No database system today performs minimization

• Reason:
  – It applies only to SELECT-DISTINCT queries (doesn’t work for bag semantics)
  – Users rarely write non-minimal queries

• However, non-minimal queries arise when using views intensively
Query Minimization for Views

An Employee is a “happy boater” if she is a boater and her manager is also a boater:

```
CREATE VIEW HappyBoaters

SELECT DISTINCT E1.name, E1.manager
FROM Employee E1, Employee E2
WHERE E1.manager = E2.name
    and E1.boater= 'YES'
    and E2.boater= 'YES'
```

This query is minimal
Query Minimization for Views

Now compute the Very-Happy-Boaters

```
SELECT DISTINCT H1.name
FROM HappyBoaters H1, HappyBoaters H2
WHERE H1.manager = H2.name
```
Query Minimization for Views

Now compute the Very-Happy-Boaters

```sql
SELECT DISTINCT H1.name
FROM HappyBoaters H1, HappyBoaters H2
WHERE H1.manager = H2.name
```

After expanding HappyBoaters, the query is not minimal
Query Minimization for Views

Now compute the Very-Happy-Boaters

```sql
SELECT DISTINCT H1.name
FROM HappyBoaters H1, HappyBoaters H2
WHERE H1.manager = H2.name
```

After expanding HappyBoaters, the query is not minimal

```sql
SELECT DISTINCT E1.name
FROM Employee E1, Employee E2, Employee E3, Employee E4
WHERE E1.manager = E2.name and E1.boater = 'YES' and E2.boater = 'YES'
    and E3.manager = E4.name and E3.boater = 'YES' and E4.boater = 'YES'
    and E1.manager = E3.name
```

E2 is redundant
Beyond CQ

- Any static analysis on relational queries (including containment, equivalence) is undecidable

- This follows from Traktenbrot’s theorem, discussed next

- Similar to Rice’s undecidability theorem for Turning machines
Trakhtenbrot’s Theorem

**Definition** A sentence $\varphi$, is called *finitely satisfiable* if there exists a finite database instance $D$ s.t. $D \models \varphi$.

**Theorem** [Trakhtenbrot] The following is undecidable: Given FO sentence $\varphi$, check if $\varphi$ is finitely satisfiable.
Trakhtenbrot’s Theorem

**Definition** A sentence $\varphi$, is called *finitely satisfiable* if there exists a finite database instance $D$ s.t. $D \models \varphi$

**Theorem** [Trakhtenbrot] The following is **undecidable**: Given FO sentence $\varphi$, check if $\varphi$ is finitely satisfiable

Is this satisfiable?

$\exists x \exists y \ R(x,y) \land \\
\forall y \ (\exists x R(x,y) \rightarrow \exists z R(y,z)) \land \\
\forall x \ \forall y \ \forall z \ (R(x,y) \land R(x,z) \rightarrow y=z)$
Trakhtenbrot’s Theorem

**Definition** A sentence $\varphi$, is called **finitely satisfiable** if there exists a finite database instance $D$ s.t. $D \models \varphi$

**Theorem** [Trakhtenbrot] The following is **undecidable**: Given FO sentence $\varphi$, check if $\varphi$ is finitely satisfiable

Is this satisfiable?

$\exists x \exists y \ R(x,y) \land \forall y \ (\exists x R(x,y) \rightarrow \exists z R(y,z)) \land \forall x \forall y \forall z \ (R(x,y) \land R(x,z) \rightarrow y=z)$

Is this satisfiable?

$\exists x \exists y \ R(x,y) \land \forall y \ (\exists x R(x,y) \rightarrow \exists z R(y,z)) \land \forall x \forall y \forall z \ (R(x,y) \land R(x,z) \rightarrow y=z) \land \exists x \ (\exists y R(x,y) \land \neg (\exists u R(u,x)))$
Query Containment

**Theorem** Query containment for Relational Calculus is **undecidable**

**Proof:** By reduction from the finite satisfiability problem:

Given a sentence $\phi$, define two queries:

$q_1 = \exists x (R(x) \land \phi)$, and $q_2 = \exists x (R(x) \land x \neq x)$

Then $q_1 \subseteq q_2$ iff $\phi$ is not finitely satisfiable
Summary of DB Theory

• We have discussed query complexity and static analysis for conjunctive queries

• Why we care:
  – Complexity helps us understand the tradeoffs between what we can express in a language and how difficult it is to implement
  – Static analysis is critical in optimizing complex queries

• Next time: transactions