CSE 544: Lecture 8

Theory

Review of Relational Algebra

- Five basic RA operators:
  - $\cup$, -, σ, π, $\times$

Selection Example

Employee

SSN	Name	DepartmentID	Salary
999999999	John	1	30,000
777777777	Tony	1	32,000
888888888	Alice	2	45,000

Find all employees with salary more than $40,000.

$\sigma_{\text{Salary} > 40000}$ (Employee)

Projection Example

Employee

SSN	Name	DepartmentID	Salary
999999999	John	1	30,000
777777777	Tony	1	32,000
888888888	Alice	2	45,000

$\Pi_{\text{SSN, Name}}$ (Employee)

Cartesian Product Example

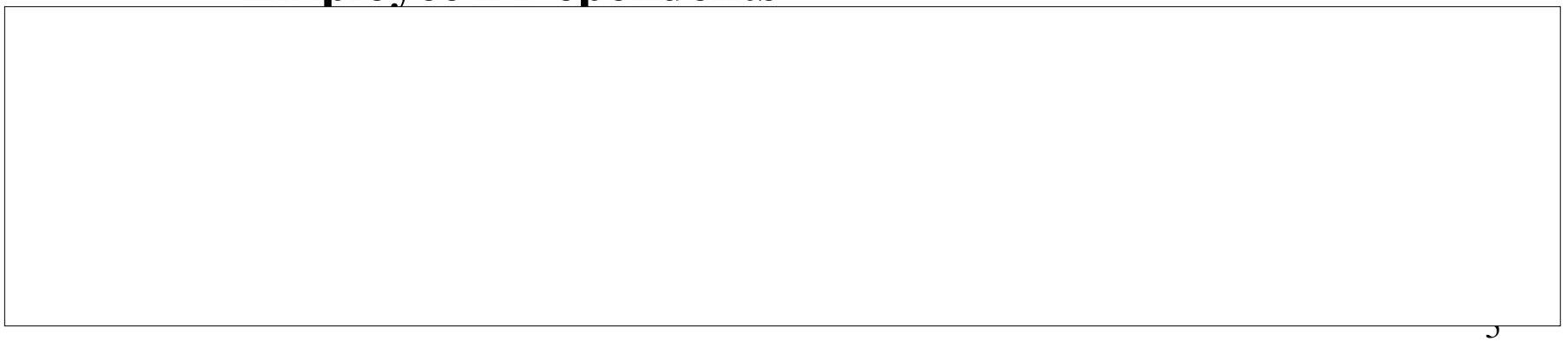
Employee

Name	SSN
John	999999999
Tony	777777777

Dependents

EmployeeSSN	Dname
999999999	Emily
777777777	Joe

Employee x Dependents

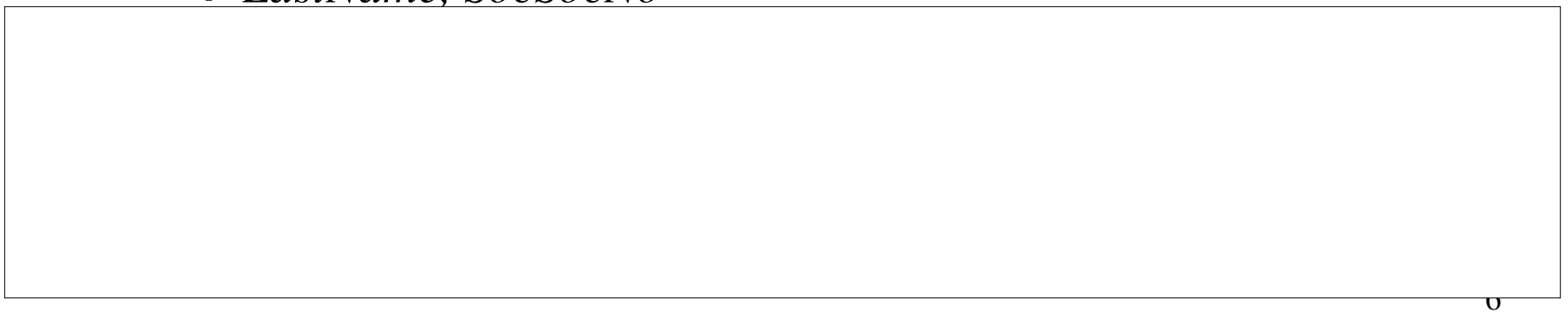


Renaming Example

Employee

Name	SSN
John	999999999
Tony	777777777

$\rho_{LastName,\ SocSocNo}$ **(Employee)**



Natural Join Example

Employee

Name	SSN
John	999999999
Tony	777777777

Dependents

SSN	Dname
999999999	Emily
777777777	Joe

Employee $\bowtie$ **Dependents =**

$$\Pi_{\text{Name, SSN, Dname}}(\sigma_{\text{SSN=SSN2}}(\text{Employee} \times \rho_{\text{SSN2, Dname}}(\text{Dependents}))$$

Natural Join

- R=

A	B
X	Y
X	Z
Y	Z
Z	V

S=

B	C
Z	U
V	W
Z	V

- R ⋈ S=

A	B	C

Natural Join

- Given the schemas R(A, B, C, D), S(A, C, E), what is the schema of R ⋈ S ?
- Given R(A, B, C), S(D, E), what is R ⋈ S ?
- Given R(A, B), S(A, B), what is R ⋈ S ?

Today's Outline

- First Order Logic as a Query Language
  - Reading assignment: 4.3
- Query languages and Complexity Classes
- Conjunctive queries

First Order Logic: Syntax

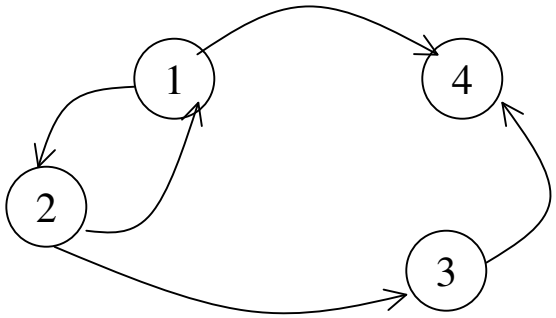
- Given:
  - A vocabulary: $R_1, \ldots, R_k$
  - An arity, $ar(R_i)$, for each $i=1,\ldots,k$
  - An infinite supply of variables $x_1, x_2, x_3, \ldots$
  - Constants: $c_1, c_2, c_3, \ldots$
- FO formulas, φ, are:

$$\varphi ::= R(t_1, \ldots, t_{ar(R)}) \mid t_i = t_j \mid \varphi \wedge \varphi' \mid \varphi \vee \varphi' \mid \neg\varphi'$$
$$\forall x.\varphi \mid \exists x.\varphi$$
$$t ::= x \mid c$$

Examples of Formulas

Most interesting case:

Vocabulary = one binary relation R (encodes a graph)



R=

1	2
2	1
2	3
1	4
3	4

Examples of Closed Formulas

- Does there exists a loop in the graph ?

$$\varphi \equiv \exists x.R(x,x)$$

- Are there paths of length >2 ?

$$\varphi \equiv \exists x.\exists y.\exists z.\exists u.(R(x,y) \wedge R(y,z) \wedge R(z,u))$$

- Is there a “sink” node ?

$$\varphi \equiv \exists x.\forall y.R(x,y)$$

Examples of Closed Formulas

- Is there a clique of size 4 ?

$$\varphi \equiv \exists x_1.\ \exists x_2.\ \exists x_3.\ \exists x_4.\ (x1 \neq x2 \wedge x1 \neq x3 \wedge \ldots \wedge x3 \neq x4 \wedge R(x_1,x_2) \wedge R(x_1,x_3) \wedge \ldots \wedge R(x_3,x_4))$$

  - Here $x1 \neq x2$ stands for $\neg x1 = x2$, etc.
- Is the graph transitively closed ?

$$\varphi \equiv \forall x.\forall y.\forall z.(R(x,y) \wedge R(y,z) \Rightarrow R(x,z))$$

  - Here $A \Rightarrow B$ stands for $\neg A \vee B$

Examples of Open Formulas

- Find all nodes connected by a path of length 2:

$$\varphi(x,y) \equiv \exists u.(R(x,u) \wedge R(u,y))$$

- Find all nodes without outgoing edges:

$$\varphi(x) \equiv \forall y.\neg R(x,y)$$

More Examples

- Vocabulary (= schema):

  Employee(name, office, mgr)

  Manager(name, office)

- Queries:
  - Find offices: $\varphi(y) \equiv (\exists x.\exists z.\text{Employee}(x,y,z) \vee \exists x.\text{Manager}(x,y))$
  - Find offices with at least two employees:

    $\varphi(y) \equiv \exists x.\exists z.\exists x'.\exists z'.(\text{Employee}(x,y,z) \wedge \text{Employee}(x',y,z') \wedge x \neq x')$
  - Find managers that share office with all their employees:
    [to do in class]

First Order Logic: Semantics

- Given a vocabulary $R_1, \ldots, R_k$
- A model is $\mathbf{D} = (D, R_1^D, \ldots, R_k^D)$
  - D = a set, called domain, or universe
  - $R_i^D \subseteq D \times D \times \ldots \times D$, (ar($R_i$) times) i = 1,...,k

First Order Logic: Semantics

- Given:
  - A model $\mathbf{D} = (D, R_1^D, ..., R_k^D)$
  - A formula φ
  - A substitution $s : \{x_1, x_2, ...\} \rightarrow D$
- We define next the relation:

$$\mathbf{D} \models \varphi[s]$$

meaning "D satisfies with s"

First Order Logic: Semantics

$\mathbf{D} \models (R(t_1, ..., t_n))\ [s]$ If $(s(t_1), ..., s(t_n)) \in R^D$

$\mathbf{D} \models (t = t')\ [s]$ If $s(t) = s(t')$

First Order Logic: Semantics

$\mathbf{D} \models (\varphi \wedge \varphi')\ [s]$ If $\mathbf{D} \models (\varphi)[s]$ and $\mathbf{D} \models (\varphi')\ [s]$

$\mathbf{D} \models (\varphi \vee \varphi')\ [s]$ If $\mathbf{D} \models (\varphi)[s]$ or $\mathbf{D} \models (\varphi')\ [s]$

$\mathbf{D} \models (\neg\varphi')\ [s]$ If not $\mathbf{D} \models (\varphi)[s]$

First Order Logic: Semantics

$\mathbf{D} \models (\forall x.\varphi)\ [s]$ If for all s' s.t. s(y) = s'(y) for all variables y other than x, $\mathbf{D} \models (\varphi)[s']$

$\mathbf{D} \models (\exists x.\varphi)\ [s]$ If for some s' s.t. s(y) = s'(y) for all variables y other than x, $\mathbf{D} \models (\varphi)[s']$

FO and Databases

FO	**Databases**
Vocabulary: $R_1, ..., R_n$	Database schema: $R_1, ..., R_n$
Model: $\mathbf{D} = (D, R_1^D, \ldots, R_k^D)$	Database instance: $\mathbf{D} = (D, R_1^D, \ldots, R_k^D)$
Formulas are *true* or *false*	Formulas compute *queries*

FO and Databases

- FO:
  a closed formula φ is *true* in **D** if **D** $\models$ φ

- Databases:
  a formula φ with free variables x_1, ..., x_n
  defines the query:

$$\varphi(\mathbf{D}) = \{(s(x_1), ..., s(x_n)) \mid \mathbf{D} \models \varphi[s]\}$$

FO and Databases

- The Relational Calculus
  - The query language consisting of FO
- The Tuple Calculus
  - A minor variation on the relational calculus
  - Uses *tuple variables* instead of *atomic variables*
  - Reading assignment 4.3
- But some "queries" in these languages make no sense
  - Define *safe* queries next

Safe Queries

A model $\mathbf{D} = (D, R_1^D, \ldots, R_k^D)$

- In FO:
  - both D and $R_1^D, \ldots, R_k^D$ may be infinite

- In databases:
  - D may infinite (**int**, **string,** etc)
  - $R_1^D, \ldots, R_k^D$ are always finite
  - We call this a *finite model*

Safe Queries

- φ is a *finite* query if for every finite model **D**, φ(**D**) is finite

- φ is a *domain independent* query if for every two finite models **D**, **D'** having the same relations:
  D = (D, R_1^D, …, R_k^D), **D'** = (D', R_1^D, …, R_k^D)
  we have φ(**D**) = φ(**D'**)

- Domain independent query aka safe query
  - Notice: book has different but equivalent definition

Unsafe Relational Queries

- Find all nodes that are not in the graph:

$$q(x) = \neg\exists y.R(x, y) \wedge \neg\exists z.R(z, x)$$

- Find all nodes that are connected to "everything":

$$q(x) = \forall y.R(x, y)$$ ←Finite, but not safe

- Find all pairs of employees or offices:

$$q(x, y) = \text{Emp}(x) \vee \text{Office}(y)$$

- We don't want such queries !

Safe Queries

- **Definition**. Given $\mathbf{D} = (D, R_1^D, \ldots, R_k^D)$, the *active domain* is D_a = the set of all constants in $R_1^D, \ldots, R_k^D$

- **Example**. Given a graph $\mathbf{D} = (D, R)$

  $$D_a = \{ x \mid \exists y.R(x,y) \vee \exists z.R(z,x)\}$$

- **Property**. If a query is safe, it suffices to range quantifiers only over the active domain (why ?)
- Hence we can *compute* safe queries

Safe Queries

- The **safe relational calculus** consists only of safe queries. However:
- **Theorem** It is undecidable if a given a FO query is safe.
- Work around:
  - Define a subset of FO queries that we *know* are safe = range restricted queries (rr-query)
  - Any safe query is equivalent to some rr-query

Range-restriction

- A syntactic condition on queries
- See [AHU] for a definition. The intuition:

Range Restricted Queries	Non-Range Restricted Queries
$S(x) \wedge \neg R(x,x)$	$\neg R(x,x)$
$S(x) \vee T(x)$	$S(x) \vee T(y)$
$\exists x.(S(x) \wedge \neg R(x,x))$	$\exists x.(\neg R(x,x))$
$\forall x.(S(x) \Rightarrow R(x,x))$	$\forall x.(R(x,x))$

Range-restriction

- **Theorem**. Every safe query is equivalent to a range-restricted query

- "Proof". Translate as follows:

$\neg R(x, y, ...)$ ➔ $x \in D_a \wedge y \in D_a \wedge .... \wedge \neg R(x, y, ...)$

$\exists x.\varphi$ ➔ $\exists x.(x \in D_a \wedge \varphi)$

$\forall x.\varphi$ ➔ $\forall x.(x \in D_a \Rightarrow \varphi)$

- From now on we assume that all queries are safe

FO = Relational Algebra

- Recall the 5 operators in the relational algebra:
  - $\cup$, -, $\times$, σ, Π
- **Theorem**. A query can be defined in the safe relational calculus iff it can be defined in the relational algebra

FO = Relational Algebra

- Proof [in class]

Limited Expressive Power

- Vocabulary: binary relation R
- The following queries cannot be expressed in FO:

- Transitive closure:
  - $\forall x.\forall y.$ there exists $x_1, ..., x_n$ s.t.
    $R(x,x_1) \wedge R(x_1,x_2) \wedge ... \wedge R(x_{n-1},x_n) \wedge R(x_n,y)$
- Parity: the number of edges in R is even

Extensions of FO

FO(LFP) = FO extended with *least fixpoint:*

- **Example**: define transitive closure like:

  $T(x,y) = R(x,y) \vee \exists z.(R(x,z) \wedge T(z,y))$

- **Meaning**. Define:

  $T_0 := \emptyset$

  $T_{n+1} = \{(x,y) \mid R(x,y) \vee \exists z.(R(x,z) \wedge T_n(z,y))\}$

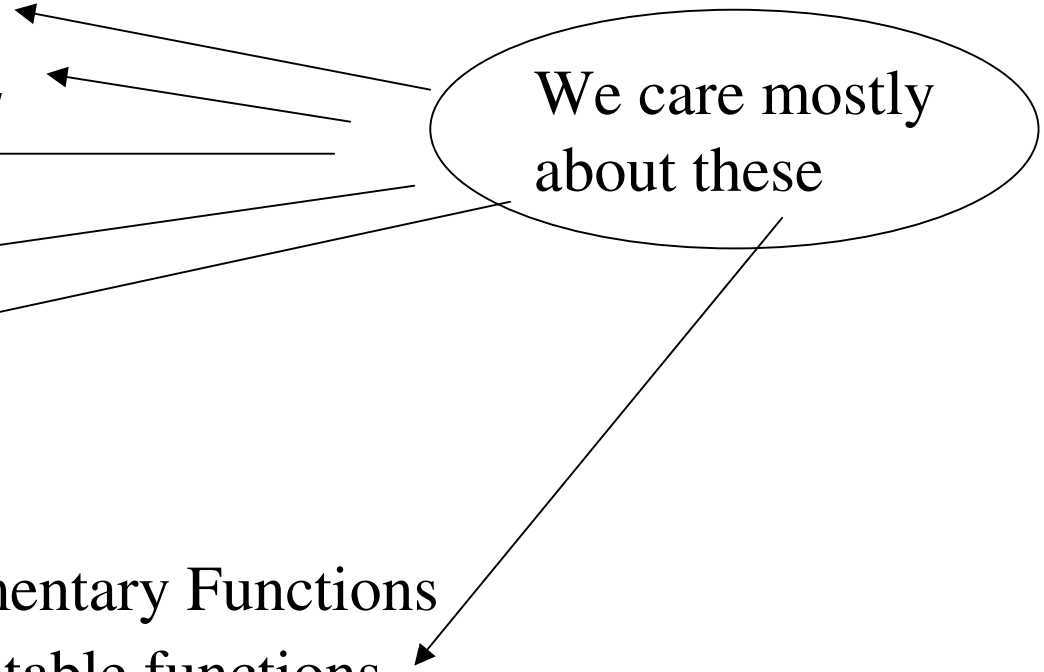
- $T_0 \subset T_1 \subset T_2 \subset \ldots \subset T_{k-1} = T_k$ stop.
- The answer is: T_k

- Q: How many steps do we need in the iteration ?

Computational Complexity Classes

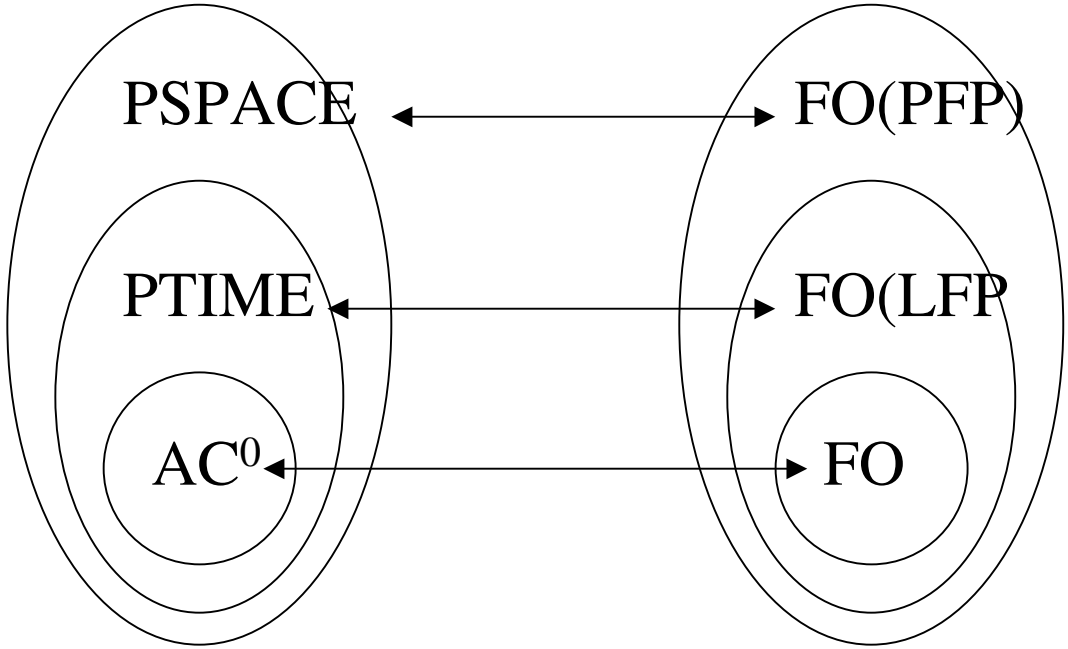
Recall computational complexity classes:

- AC^0
- LOGSPACE
- NLOGSPACE
- PTIME
- NP
- PSPACE
- EXPTIME
- EXPSPACE
- (Kalmar) Elementary Functions
- Turing Computable functions



Query Languages and Complexity Classes

Paper: On the Unusual Effectiveness of Logic in Computer Science



Important: the more complex a QL, the harder it is to optimize

Conjunctive Queries

- **Definition** A conjunctive query is a FO restricted to $R(t_1, ..., t_n)$, $\exists$, $\wedge$
  (missing are $\forall$, $\vee$, $\neg$)
- CQ = all conjunctive queries

- Any CQ query can be written as:
  $\exists x_1. \exists x_2... \exists x_n.(R_1(t_{11},...,t_{1m}) \wedge ... \wedge R_k(t_{k1},...,t_{km}))$
- Same in **Datalog** notation:
  $A(x_1,...,x_n)$:- $R_1(t_{11},...,t_{1m})$, ... , $R_k(t_{k1},...,t_{km})$

Examples

Employee(x), ManagedBy(x,y), Manager(y)

- Find all employees having the same manager as Smith:

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

Examples

Employee(x), ManagedBy(x,y), Manager(y)

- Find all employees having the same director as Smith:

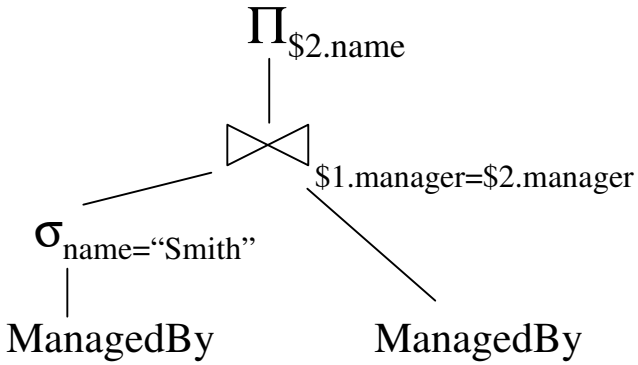
  A(x) :- ManagedBy(“Smith”,y), ManagedBy(y,z),
  ManagedBy(x,u), ManagedBy(u,z)

Equivalent Formulations

Relational Algebra:

- Conjunctive queries correspond precisely to σ_C, Π_A, $\times$ (missing: $\cup$, $-$)

- A(x) :- ManagedBy("Smith",y), ManagedBy(x,y)



Equivalent Formulations

SQL:

- Conjunctive queries correspond to single select-disticnt-from-where blocks with equality conditions in the WHERE clause

```
select distinct m2.name
from ManagedBy m1, ManagedBy m2
where m1.name="Smith" AND
        m1.manager=m2.manager
```

Conjunctive Queries

- Most useful class of queries
- Also enjoys remarkable, positive properties

- Focus of research during 70's, 80's
- Still focus of research in the 00's

- We discuss the most celebrated property of conjunctive queries: containment is decidable

Query Containment

- **Definition** Given two queries q_1, q_2, we say that q_1 is contained in q_2 if for every database **D**, $q_1(\mathbf{D}) \subseteq q_2(\mathbf{D})$.

- Notation: $q_1 \subseteq q_2$

- Obviously: if $q_1 \subseteq q_2$ and $q_2 \subseteq q_1$ then $q_1 = q_2$.

Examples of Query Containments

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

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

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

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

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

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

In all cases:
$q_1 \subseteq q_2$

Examples of Query Containments

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

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

Then $q_1 \subseteq q_2$ (why ?)

Examples of Query Containments

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

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

Then $q_1 \subseteq q_2$ (why ?)

Query Containment

- **Theorem** Query containment for FO is undecidable

- **Theorem** Query containment for CQ is decidable and NP-complete.

Query Containment

The most interesting part: *how* we check $q_1 \subseteq q_2$

- The *canonical database* and the *canonical tuple* for q_1:
  - Canonical database: $\mathbf{D_{q1}} = (D, R_1, \ldots, R_k)$ where:
    - D = all variables and constants in q_1
    - $R_1, \ldots, R_k$ = the body of q_1
  - Canonical tuple: t_{q1} = the head of q_1

Examples of Canonical Databases

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

- $\mathbf{D}_{q1} = (D, R)$
  - D={x,y,u,v}
  - R =

x	y
v	u
v	y

- $t_{q1} = (x,y)$

Examples of Canonical Databases

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

- $\mathbf{D}_{q1}$ = (D, R)
  - D={x,u,"Smith","Fred"}
  - R =

x	u
u	"Smith"
u	"Fred"
u	u

- t_{q1} = (x)

Checking Containment

Theorem: $q_1 \subseteq q_2$ iff $t_{q1} \in q_2(\mathbf{D}_{q1})$.

Example:

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

- D={x,y,u,v}
- R =

x	y
v	u
v	y

$t_{q1} = (x,y)$

- Yes, $q_1 \subseteq q_2$

Query Homeomorphisms

- How do we *evaluate* q_2 on $\mathbf{D}_{q1}$?

- A ***homeomorphism*** f : $q_2 \rightarrow q_1$ is a function
  f: var(q_2) → var(q_1) U const(q_1)
  such that:
  - f(body(q_2)) ⊆ body(q_1)
  - f(canonicalTuple(q_2)) = canonicalTuple(q_1)

Example of Query Homeomorphism

$var(q_1) = \{x, u, v, y\}$

$var(q_2) = \{x, u, v, w, t, y\}$

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

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

Example of Query Homeomorphism

$var(q_1)$ U $const(q_1)$ = {x,u, “Smith”}

$var(q_2)$ = {x,u,v,w}

$q_1(x)$:- R(x,u), R(u,”Smith”), R(u,”Fred”), R(u, u)

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

The Homeomorphism Theorem

- ***Theorem*** $q_1 \subseteq q_2$ iff there exists a homeomorphism from q_2 to q_1.

- ***Theorem*** Conjunctive query containment is:
  (1) decidable (why ?)
  (2) in NP (why ?)
  (3) NP-hard
- Short: it is NP-complete