CSE 544: Lecture 9

Conjunctive Queries, Views, Datalog

Monday, 4/29/2002

Conjunctive Queries

- A **conjunctive query** is an FO formula containing:
  - $R(t_1, ..., t_n)$, $\exists$, $\wedge$

  (missing are $\forall$, $\vee$, $\neg$)

- CQ = set of conjunctive queries

- Example: $q(x,y) = \exists z.(R(x,z) \wedge \exists u.(R(z,u) \wedge R(u,y)))$

Conjunctive Queries

- Any CQ query can be written as:

$$q(x_1,...,x_n) = \exists y_1.\ \exists y_2 ...\ \exists y_p.(R_1(t_{11},...,t_{1m}) \wedge ... \wedge R_k(t_{k1},...,t_{km}))$$

(why ?)

- Same in **Datalog** notation:

Datalog rule

$$q(x_1,...,x_n) :- R_1(t_{11},...,t_{1m}),\ ...\ ,\ R_k(t_{k1},...,t_{km}))$$

head

body

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:

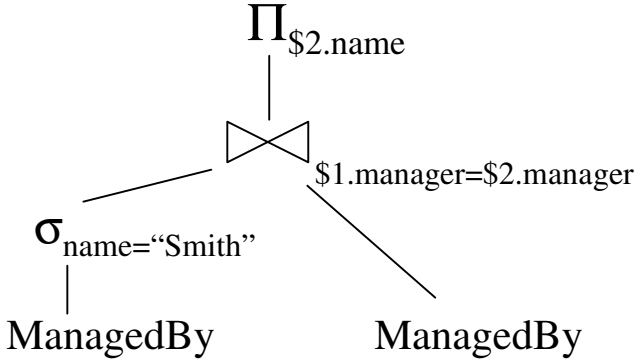
```
A(x) :- ManagedBy("Smith",y), ManagedBy(y,z),
        ManagedBy(x,u), ManagedBy(u,z)
```

CQ and Relational Algebra

Relational Algebra:

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

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



CQ and SQL

SQL:

- Conjunctive queries correspond to single select-distinct-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

- Main focus of optimization techniques

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

- Properties of CQ:
  - Containment is decidable [Chandra&Merlin'77]
  - Query rewriting using views [Levy et al.'95, Ullman'99]

Query Containment

- Query 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

In which cases is q1 $\subseteq$ q2 ?

```
q1(x) :- R(x,u), R(u,v), R(v,w)
q2(x) :- R(x,u), R(u,v)
```

```
q1(x) :- R(x,u), R(u,v), R(v,x)
q2(x) :- R(x,u), R(u,x)
```

```
q1(x) :- R(x,u), R(u,u)
q2(x) :- R(x,u), R(u,v)
```

```
q1(x) :- R(x,u), R(u,"Smith")
q2(x) :- R(x,u), R(u,v)
```

Query Containment

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

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

Query Containment Algorithm

How to check $q_1 \subseteq q_2$

- **Canonical database** for q_1 is:

  $$\mathbf{D_{q1}} = (D, R_1^D, \ldots, R_k^D)$$

  - D = all variables and constants in q_1
  - $R_1^D, \ldots, R_k^D$ = the body of q_1
- **Canonical tuple** for q_1 is:

  t_{q1} (the head of q_1)

Examples of Canonical Databases

q1(x,y) :- R(x,u),R(v,u),R(v,y)

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

x	u
v	u
v	y

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

Examples of Canonical Databases

q1(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	u
v	u
v	y

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

- Yes, $q_1 \subseteq q_2$

Query Homomorphisms

- A ***homomorphism*** $f : q_2 \rightarrow q_1$ is a function $f: var(q_2) \rightarrow var(q_1) \cup const(q_1)$ such that:
  - $f(body(q_2)) \subseteq body(q_1)$
  - $f(t_{q1}) = t_{q2}$

The Homomorphism Theorem $q_1 \subseteq q_2$ iff there exists a homomorphism $f : q_2 \rightarrow q_1$

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

$var(q_1) \cup const(q_1) = \{x,u, \text{“Smith”}\}$

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

$q_1(x)$:- 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

Views

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

Views

L(x,y) :- ManagedBy(x,u), ManagedBy(u,y)

E(x,y) :- ManagedBy(x,y), Employee(y)

Query

Q(x,y) :- ManagedBy(x,u), ManagedBy(u,v),
ManagedBy(v,w), ManagedBy(w,y), Employee(y)

How can we answer Q if we only have L and E ?

Views

- Query rewriting using views (when possible):

```
Q(x,y) :- L(x,u), L(u,y), E(v,y)
```

- Query answering:
  - Sometimes we cannot express it in CQ or FO, but we can still answer it

Views

Applications:

- Using advanced indexes
- Using replicated data
- Data integration [Ullman'99]

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

Datalog

- Adds recursion, so we can compute transitive closure
- A datalog program (query) consists of several datalog rules:

$$P_1(t_1) \text{ :- } body_1$$
$$P_2(t_2) \text{ :- } body_2$$
$$\ldots$$
$$P_n(t_n) \text{ :- } body_n$$

Datalog

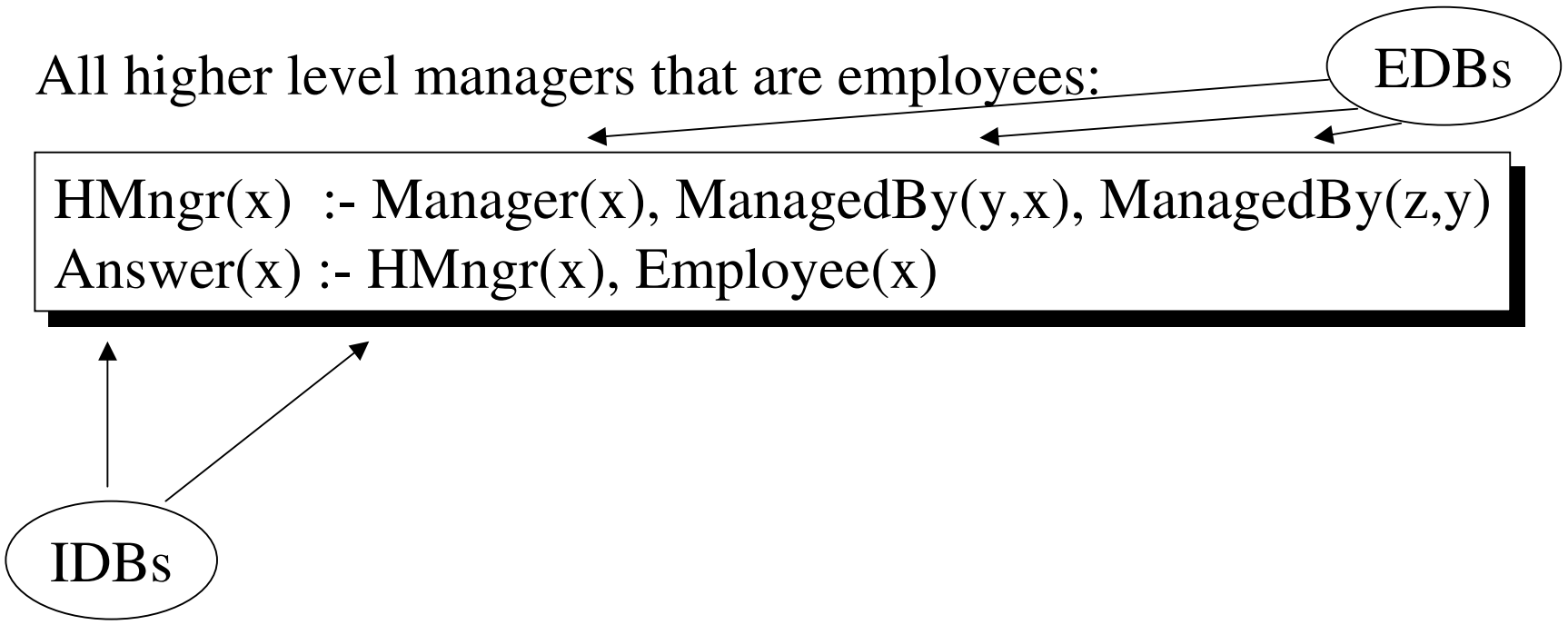
Terminology:

- EDB = extensional database predicates
  - The database predicates
- IDB = intentional database predicates
  - The new predicates constructed by the program

Datalog

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

All higher level managers that are employees:



Datalog

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

All persons:

```
Person(x) :- Manager(x)
Person(x) :- Employee(x)
```

Manger $\cup$ Employee

Datalog

Graph: R(x,y)

```
P(x,y) :- R(x,u), R(u,v), R(v,y)
A(x,y) :- P(x,u), P(u,y)
```

Can "unfold" it into:

```
A(x,y) :- R(x,u), R(u,v), R(v,w), R(w,m), R(m,n), R(n,y)
```

Recursion in Datalog

Graph: R(x,y)

Transitive closure:

```
P(x,y) :- R(x,y)
P(x,y) :- P(x,u), R(u,y)
```

Transitive closure:

```
P(x,y) :- R(x,y)
P(x,y) :- P(x,u), P(u,y)
```

Recursion in Datalog

Boolean trees:

Leaf0(x), Leaf1(x),
AND(x, y_1, y_2), OR(x, y_1, y_2),
Root(x)

- Find out if the tree value is 0 or 1

```
One(x)      :- Leaf1(x)
One(x)      :- AND(x, y1, y2), One(y1), One(y2)
One(x)      :- OR(x, y1, y2), One(y1)
One(x)      :- OR(x, y1, y2), One(y2)
Answer()    :- Root(x), One(x)
```

Exercise

Boolean trees:

Leaf0(x), Leaf1(x),
AND(x, y_1, y_2), OR(x, y_1, y_2), Not(x,y),
Root(x)

- **Hint**: compute both One(x) and Zero(x)
  here you need to use Leaf0

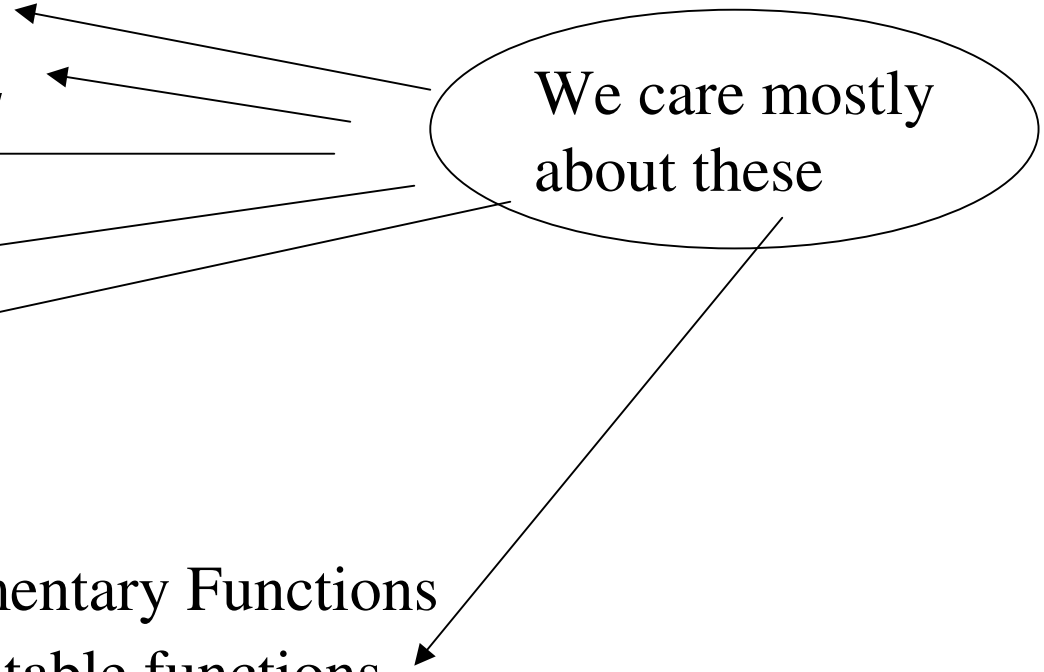
Variants of Datalog

	without recursion	with recursion
without ¬	Non-recursive Datalog = union of CQ (why ?)	**Datalog**
with ¬	Non-recursive Datalog¬ = FO	Datalog¬

Computational Complexity Classes

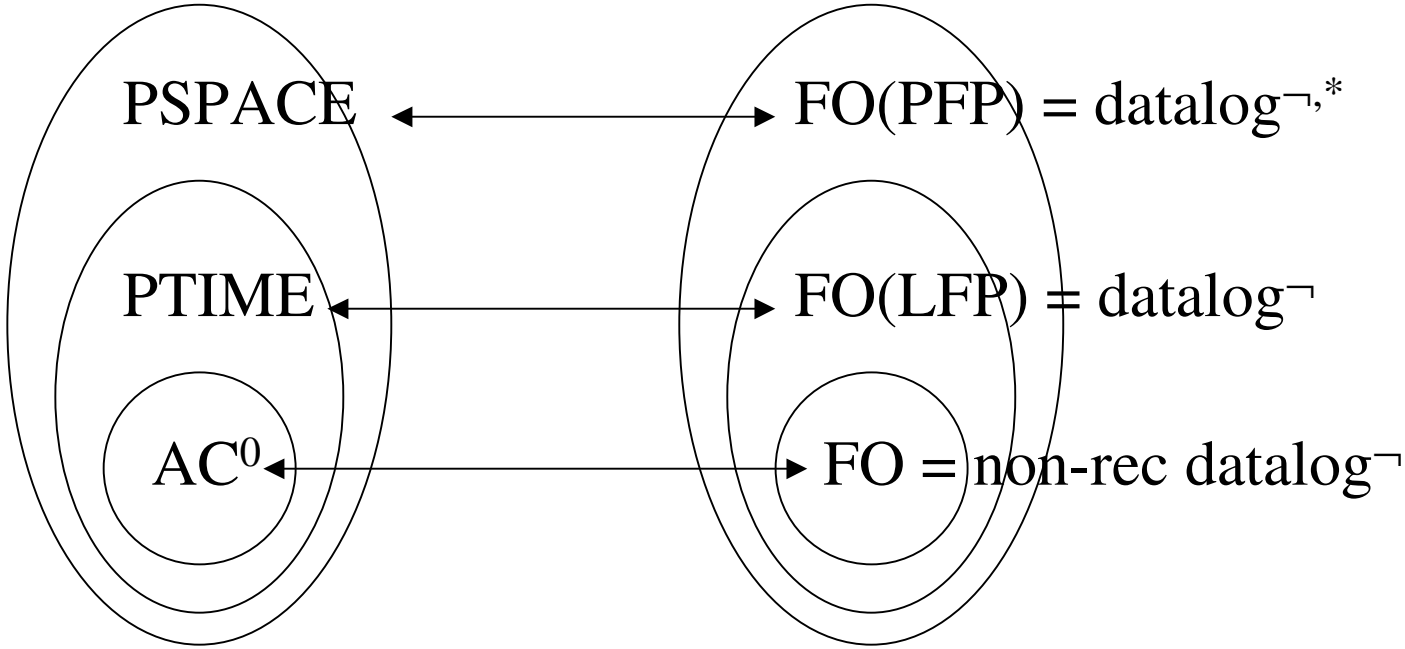
Recall computational complexity classes:

- 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