Database Theory: Beyond FO

CS 645
Mar 6, 2006
Coming lectures

• TODAY:
  – Review of Containment for CQs
  – Containment beyond CQs
    • extensions to CQs, FO queries
  – Limited expressiveness of FO
  – Adding recursion (Datalog)

• Next Class
  – Expressiveness & Complexity
  – Views & Reasoning about Views
Conjunctive Queries

Intuitively: when facts in the body are true of stored relations, then we infer the fact in the head

\[ P(x,z) :- R(x,y) \& R(y,z) \]

- More formally:
- Consider all possible **substitutions**: assignments of the variables in the body
Properties of Conjunctive Queries

• Satisfiability
  – A query $q$ is **satisfiable** if there exists some input relation $I$ such that $q(I)$ is non-empty.
  – **FACT:** Every CQ is satisfiable.

• Monotonicity
  – A query $q$ is **monotonic** if for each instance $I,J$ over schema, $I \subseteq J$ implies $q(I) \subseteq q(J)$.
  – **FACT:** Every CQ is monotonic.
Query language classes

**Recursive Queries**

**FO queries**

**Conjunctive Queries**

- RA
- (safe) RC
- UCQ
- SFW +
  - UNION
  - EXCEPT

- RA: $\sigma, \pi, x$
- single datalog rule
- $S^dFW$
Query Equivalence and Containment

• One kind of static analysis
• Useful for query optimization
• Global v. local optimization
Query Containment

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

• Notation: $q_1 \subseteq q_2$

• Obviously: $q_1 \subseteq q_2$ and $q_2 \subseteq q_1$ iff $q_1 \equiv q_2$

• Conversely: $q_1 \land q_2 \equiv q_2$ iff $q_1 \subseteq q_2$

We will study the containment problem only.
Checking containment

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

1. “Freeze” q1
   - Replace variables by unique constants
   - \( x \rightarrow a_x \), \( u \rightarrow a_y \)
   - this is called canonical database of q1

2. Evaluate q2 on frozen body of q1

3. If frozen head is derived, then
   \( q_1 \subseteq q_2 \)

Containment!
<table>
<thead>
<tr>
<th>Examples of Query Containment</th>
<th>Is $q_1 \subseteq q_2$?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_1(x) :- R(x,y), R(y,z), R(z,w)$</td>
<td>$q_1(x) :- R(x,y), R(y,z), R(z,x)$</td>
</tr>
<tr>
<td>$q_2(x) :- R(x,y), R(y,z)$</td>
<td>$q_2(x) :- R(x,y), R(y,x)$</td>
</tr>
<tr>
<td>$q_1(x) :- R(x,u), R(u,u)$</td>
<td>$q_1(x) :- R(x,u), R(u,&quot;Smith&quot;)$</td>
</tr>
<tr>
<td>$q_2(x) :- R(x,u), R(u,v), R(v,w)$</td>
<td>$q_2(x) :- R(x,u), R(u,v)$</td>
</tr>
</tbody>
</table>
Query Homomorphisms

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

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

Chandra & Merlin 1977
The Homeomorphism Theorem

• **Theorem** Conjunctive query containment is:
  1. decidable (why ?)
  2. in NP (why ?)
  3. NP-hard

• In short: containment for CQs is NP-complete
Query Minimization

**Definition** A conjunctive query $q$ is *minimal* if for every other conjunctive query $q'$ s.t. $q \equiv q'$, $q'$ has at least as many predicates (‘subgoals’) as $q$.

Are these queries minimal?

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

• Query minimization algorithm

  1. Choose a subgoal $g$ of $q$
  2. Remove $g$: let $q'$ be the new query
     - We already know $q \subseteq q'$ (why ?)
  3. If $q' \subseteq q$ then permanently remove $g$

• The order in which we inspect subgoals doesn’t matter
Other containment problems

• Extensions of CQs:
  – Unions of CQs
  – CQs with inequality

• FO queries

• Containment under constraints

• What about bags?
  – strange things happen
Query Containment for UCQ

\[ Q = q_1 \cup q_2 \cup q_3 \cup \ldots \cup q_m \quad Q' = q_1' \cup q_2' \cup q_3' \cup \ldots \cup q_n' \]

\[ Q \subseteq Q' \text{ iff} \]
\[ q_1 \cup q_2 \cup q_3 \cup \ldots q_m \subseteq q_1' \cup q_2' \cup q_3' \cup \ldots q_n' \]

Notice: \[ q_1 \cup q_2 \cup q_3 \cup \ldots \subseteq Q' \text{ iff} \]
\[ q_1 \subseteq Q' \text{ and } q_2 \subseteq Q' \text{ and } q_3 \subseteq Q' \text{ and } \ldots \]

**Theorem** \[ q_i \subseteq q_1' \cup q_2' \cup q_3' \cup \ldots \text{ iff there exists some } k \text{ such that } q \subseteq q_k' \]

It follows that containment for UCQ is decidable, and NP-complete.
Query Containment for CQ<

$q_1(x) :- R(x,y,z)$
$q_2(x) :- R(x,y,z), y < z$

Does canonical database test work?
Is $q_1 \subseteq q_2$?

To check containment do this:
- Consider all possible partitions of variables in $q_1$, and then all ordering of assignments to the partitions.
- For each of them check containment of $q_1$ in $q_2$
- If all hold, then $q_1 \subseteq q_2$

Still decidable, but harder than NP: now in $\Pi_p^2$
Containment under constraints

• Recall: query $q_1$ is **contained** in $q_2$ if for every database $D$, $q_1(D) \subseteq q_2(D)$.

• What if we know more about our input databases?

• Replace “every database $D$”, with:
  – “every database satisfying constraint $C$”

  Containment under FD is NP-complete
Containment for FO queries

- **Theorem** Satisfiability for FO queries is undecidable
- **Lemma** Query containment/equivalence for FO is undecidable
  - if we had an algorithm for equivalence, we could use it to decide satisfiability of q:
  - check: $q \equiv \text{false}$

Consequence: we cannot do global query optimization for first-order queries.
Expressive Power of FO

• Let $I = \{R(x,y)\}$ represent a graph
• Query $\text{path}(x,y) =$
  – all $x,y$ such that there is a path from $x$ to $y$

• Theorem: $\text{path}(x,y)$ cannot be expressed in FO.
Datalog Programs

• A **Datalog program** is a collection of rules.

• In a program, subgoals can be either
  1. **EDB** = **Extensional Database** = stored table.
  2. **IDB** = **Intensional Database** = computed table.

  ▶ Never both! No EDB in heads.
Non-recursive rules

Graph: \( R(x,y) \)

\[
\begin{align*}
P(x,y) & :\! - \ R(x,u), \ R(u,v), \ R(v,y) \\
A(x,y) & :\! - \ P(x,u), \ P(u,y)
\end{align*}
\]

Can “unfold” it into:

\[
\begin{align*}
A(x,y) & :\! - \ R(x,u), \ R(u,v), \ R(v,w), \ R(w,m), \ R(m,n), \ R(n,y)
\end{align*}
\]
Example: Datalog Program

• Using EDB `Sells(bar, beer, price)` and `Beers(name, manf)`, find the manufacturers of beers Joe doesn’t sell.

```
JoeSells(b) <- Sells(‘Joes Bar’, b, p)
Answer(m) <- Beers(b,m) & ¬JoeSells(b)
```
Evaluating Datalog Programs

• As long as there is no recursion, we can pick an order to evaluate the IDB predicates, so that all the predicates in the body of its rules have already been evaluated.

• If an IDB predicate has more than one rule, each rule contributes tuples to its relation.
Expressive Power of Datalog

• Without recursion, Datalog expresses exactly the first order queries
  - negated subgoals
  - implicit union

• We call this
  - “non-recursive Datalog with negation”
  - denoted: nr-Datalog
Query language classes

Expressiveness

Recursive Queries

FO queries

Conjunctive Queries

Algebra

Logic

SQL

RA

(safe) RC

UCQ

nr-Datalog

SFW +

UNION

EXCEPT

RA:

$\sigma,\pi,x$

single
datalog
rule

$S^dFW$
Recursive example

Two forms of transitive closure:

Graph: \( R(x,y) \)

\[
\begin{align*}
\text{Path}(x,y) & :\ - R(x,y) \\
\text{Path}(x,y) & :\ - \text{Path}(x,u), \ R(u,y)
\end{align*}
\]

\[
\begin{align*}
\text{Path}(x,y) & :\ - R(x,y) \\
\text{Path}(x,y) & :\ - \text{Path}(x,u), \ \text{Path}(u,y)
\end{align*}
\]