# Conjunctive Queries

Containment Mappings

Canonical Databases

Sariaya's Algorithm

# Conjunctive Queries

- A CQ is a single Datalog rule, with all subgoals assumed to be EDB.
- *Meaning* of a CQ is the mapping from databases (the EDB) to the relation produced for the head predicate by applying that rule to the EDB.

# Containment of CQ's

- Q1 $\subseteq$ Q2 iff for all databases *D*, Q1(*D*) $\subseteq$ Q2(*D*).
- Example:
  - Q1: `p(X,Y) :- arc(X,Z) & arc(Z,Y)`
  - Q2: `p(X,Y) :- arc(X,Z) & arc(W,Y)`
- DB is a graph; Q1 produces paths of length 2, Q2 produces pairs of nodes with an arc out and in, respectively.

# Example --- Continued

- Whenever there is a path from $X$ to $Y$, it must be that $X$ has an arc out, and $Y$ an arc in.
- Thus, every fact (tuple) produced by Q1 is also produced by Q2.
- That is, Q1 $\subseteq$ Q2.

# Why Care About CQ Containment?

- Important optimization: if we can break a query into terms that are CQ's, we can eliminate those terms contained in another.
  - Especially important when we deal with integration of information: CQ containment is almost the only way to tell what information from sources we <u>don't</u> need.

# Why Care? --- Continued

- Containment tests imply equivalence-of-programs tests.
  - Any theory of program (query) design or optimization requires us to know when programs are equivalent.
  - CQ's, and some generalizations to be discussed, are the most powerful class of programs for which equivalence is known to be decidable.

# Why Care --- Concluded

- Although CQ theory first appeared at a database conference, the AI community has taken CQ's to heart.
- CQ's, or similar logics like description logic, are used in a number of AI applications.
  - Again --- their design theory is really containment and equivalence.

# Testing Containment

- Two approaches:
  1. Containment mappings.
  2. Canonical databases.
- Really the same in the simple CQ case covered so far.
- Containment is NP-complete, but CQ's tend to be small so here is one case where intractability doesn't hurt you.

# Containment Mappings

- A mapping from the variables of CQ Q2 to the variables of CQ Q1, such that:
  1. The head of Q2 is mapped to the head of Q1.
  2. Each subgoal of Q2 is mapped to some subgoal of Q1 with the same predicate.

# Important Theorem

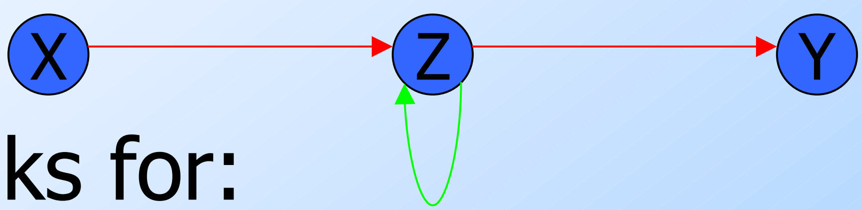
- There is a containment mapping from Q2 to Q1 if and only if Q1 $\subseteq$ Q2.
- Note that the containment mapping is opposite the containment --- it goes from the larger (containing CQ) to the smaller (contained CQ).

# Example

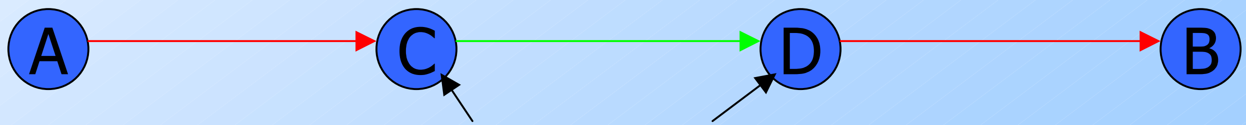
Q1: `p(X,Y):-r(X,Z) & g(Z,Z) & r(Z,Y)`

Q2: `p(A,B):-r(A,C) & g(C,D) & r(D,B)`

Q1 looks for:




Q2 looks for:




Since C=D is possible,
expect Q1 $\subseteq$ Q2.

# Example --- Continued

Q1: `p(X,Y):-r(X,Z) & g(Z,Z) & r(Z,Y)`

Q2: `p(A,B):-r(A,C) & g(C,D) & r(D,B)`

Containment mapping: m(A)=X; m(B)=Y; m(C)=m(D)=Z.

# Example ---Concluded

Q1: `p(X,Y):-r(X,Z) & g(Z,Z) & r(Z,Y)`

Q2: `p(A,B):-r(A,C) & g(C,D) & r(D,B)`

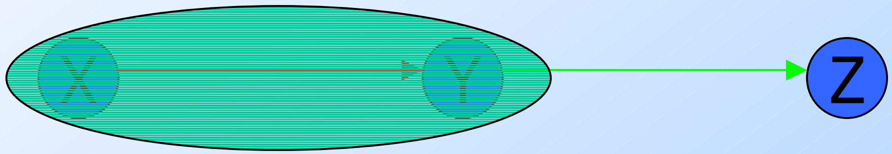
- No containment mapping from Q1 to Q2.
  - g(Z,Z) can only be mapped to g(C,D).
    - No other *g* subgoals in Q2.
  - But then Z must map to both C and D --- impossible.
- Thus, Q1 properly contained in Q2.

# Another Example

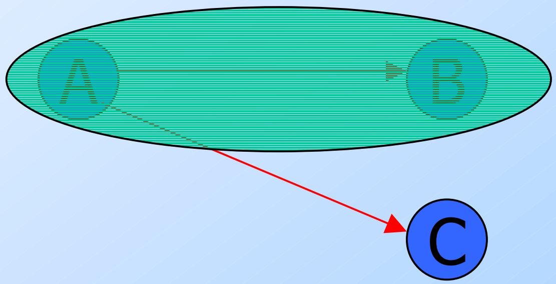
Q1: `p(X,Y):-r(X,Y) & g(Y,Z)`

Q2: `p(A,B):-r(A,B) & r(A,C)`

Q1 looks for:




Q2 looks for:



# Example --- Continued

And not every subgoal need be a target.

`Q1: p(X,Y):-r(X,Y) & g(Y,Z)`

`Q2: p(A,B):-r(A,B) & r(A,C)`

Notice two subgoals can map to one.

Containment mapping: m(A)=X;
m(B)=m(C)=Y.

# Example ---Concluded

Q1: `p(X,Y):-r(X,Y) & g(Y,Z)`
Q2: `p(A,B):-r(A,B) & r(A,C)`

- No containment mapping from Q1 to Q2.
  - g(Y,Z) cannot map anywhere, since there is no *g* subgoal in Q2.
- Thus, Q1 properly contained in Q2.

# Proof of Containment-Mapping Theorem --- (1)

- First, assume there is a CM *m* : Q2->Q1.
- Let *D* be any database; we must show that Q1(*D*) $\subseteq$ Q2(*D*).
- Suppose *t* is a tuple in Q1(*D*); we must show *t* is also in Q2(*D*).

# Proof --- (2)

- Since *t* is in Q1(*D*), there is a substitution *s* from the variables of Q1 to values that:
  1. Makes every subgoal of Q1 a fact in *D*.
     - More precisely, if p(X,Y,…) is a subgoal, then [s(X),s(Y),…] is a tuple in the relation for *p*.
  2. Turns the head of Q1 into *t*.

# Proof --- (3)

- Consider the effect of applying *m* and then *s* to Q2.

$$\begin{tikzcd}
\text{head of Q2 :-} \arrow[d, "m"'] & \text{subgoal of Q2} \arrow[d, "m"'] \arrow[dd, bend left] \\
\text{head of Q1 :-} \arrow[d, "s"'] & \text{subgoal of Q1} \arrow[d, "s"'] \\
t & \text{tuple of } D
\end{tikzcd}$$

$s \circ m$ maps each sub-goal of Q2 to a tuple of *D*.

And the head of Q2 becomes *t*, proving *t* is also in Q2(*D*); i.e., Q1 $\subseteq$ Q2.

# Proof of Converse --- (1)

- Now, we must assume Q1 $\subseteq$ Q2, and show there is a containment mapping from Q2 to Q1.
- Key idea --- <u>frozen</u> CQ *Q* :
  1. For each variable of *Q*, create a corresponding, unique constant.
  2. Frozen *Q* is a DB with one tuple formed from each subgoal of *Q*, with constants in place of variables.

# Example: Frozen CQ

```
p(X,Y):-r(X,Z) & g(Z,Z) & r(Z,Y)
```

- Let's use lower-case letters as constants corresponding to variables.
- Then frozen CQ is:

Relation $R$ for predicate $r$ = {(x,z), (z,y)}.

Relation $G$ for predicate $g$ = {(z,z)}.

# Converse --- (2)

- Suppose Q1 $\subseteq$ Q2, and let *D* be the frozen Q1.
- Claim: Q1(*D*) contains the frozen head of Q1 --- that is, the head of Q1 with variables replaced by their corresponding constants.
  - Proof: the "freeze" substitution makes all subgoals in *D*, and makes the head become the frozen head.

# Converse --- (3)

- Since Q1 $\subseteq$ Q2, the frozen head of Q1 must also be in Q2(*D*).
- Thus, there is a mapping *s* from variables of Q2 to *D* that turns subgoals of Q2 into tuples of *D* and turns the head of Q2 into the frozen head of Q1.
- But tuples of *D* are frozen subgoals of Q1, so *s* followed by "unfreeze" is a containment mapping from Q2 to Q1.

# In Pictures

Q2: h(X,Y) :- … p(Y,Z) …

s s

h(u,v) p(a,b) *D*

freeze

Q1: h(U,V) :- … p(A,B) …

*s* followed by inverse of *freeze* maps each subgoal p(Y,Z) of Q2 to a subgoal p(A,B) of Q1 and maps h(X,Y) to h(U,V).

# Dual View of CM's

- Instead of thinking of a CM as a mapping on variables, think of a CM as a mapping from atoms to atoms.
- Required conditions:
  1. The head must map to the head.
  2. Each subgoal maps to a subgoal.
  3. As a consequence, no variable is mapped to two different variables.

# Canonical Databases

- General idea: test Q1 $\subseteq$ Q2 by checking that Q1($D_1$) $\subseteq$ Q2($D_1$),…, Q1($D_n$) $\subseteq$ Q2($D_n$), where $D_1$,…,$D_n$ are the canonical databases.
- For the standard CQ case, we only need one canonical DB --- the frozen Q1.
- But in more general forms of queries, larger sets of canonical DB's are needed.

# Why Canonical DB Test Works

- Let $D$ = frozen body of Q1; $h$ = frozen head of Q1.
- Theorem: Q1 $\subseteq$ Q2 iff Q2($D$) contains $h$.
- Proof (only if): Suppose Q2($D$) does not contain $h$. Since Q1($D$) surely contains $h$, it follows that Q1 is not contained in Q2.

# Proof (if):

- Suppose Q2(*D*) contains *h*.
- Then there is a mapping from the variables of Q2 to the constants of *D* that maps:
  - The head of Q2 to *h*.
  - Each subgoal of Q2 to a frozen subgoal of Q1.
- This mapping, followed by "unfreeze," is a containment mapping, so Q1 $\subseteq$ Q2.

# Sariaya's Algorithm

- Containment of CQ's is NP-complete.
- But Sariaya's algorithm is a linear-time test for the common situation where Q1 (the contained query) has no more than two subgoals with any one predicate.
- Reduction to 2SAT.
- We'll give a simple, quadratic version.

# Saraiya’s Algorithm --- (2)

1. For any subgoal p(…) of Q2, where there is only one *p* –subgoal of Q1, we know exactly where p(…) must map.
2. If there is a subgoal of Q2 that can map to two different subgoals of Q1, assume one choice, and chase down the “consequences.”

# Consequences

1. If $p(X_1,\ldots,X_n)$ is known to map to $p(Y_1,\ldots,Y_n)$, then we know each vaiable $X_i$ maps to $Y_i$.
2. If $p(X_1,\ldots,X_n)$ is a subgoal of Q2, and we know $X_i$ maps to some variable $Z$, and only one of the $p$ –subgoals of Q1 has $Z$ in the $i$ th component, then $p(X_1,\ldots,X_n)$ must map to that subgoal.

# Sariaya's Algorithm --- (3)

- Eventually, one of two things happens:
  1. We derive a contradiction --- a subgoal or variable that must map to two different things.
  2. We close the set of inferences --- there is no contradiction, and no more consequences.

# Case (1): Contradiction

- In this case, we go back and try the other choice if there is one, and fail if there is no other choice.

# Case (2): Closure

- In this case, we have found some variables and subgoals of Q2 that can be mapped as chosen, with no effect on any remaining subgoals or variables.
- Fix these choices, and consider any remaining subgoals.
- If all subgoals are now mapped, we have found a CM and are done.

# Example

Q2: p(X) :- a(X,Y) & b(Y,Z) & b(Z,W) & a(W,X)

Q1: p(B) :- a(A,B) & a(B,A) & b(A,C) & b(C,B)

Start by choosing
a(X,Y) -> a(A,B)

Then X->A
and Y->B

Now, b(Y,Z) must
map to some b(B,?).
But both choices do
not have first com-
ponent B.

# Example --- Continued

Q2: p(X) :- a(X,Y) & b(Y,Z) & b(Z,W) & a(W,X)

Q1: p(B) :- a(A,B) & a(B,A) & b(A,C) & b(C,B)

We thus know that in any CM, a(X,Y) maps to a(B,A). Thus, X->B and Y->A.

Then b(Y,Z) must map to b(A,C), and Z->C.

Thus, b(Z,W) -> b(C,B), and W->B

a(W,X) cannot map to a(A,B) [W doesn't map to A] or to a(B,A) [X doesn't map to A]. Complete failure.

# Example ---Slight Variation

Q2: p(X) :- a(X,Y) & b(Y,Z) & b(Z,W) & a(W,X)

Q1: p(B) :- a(A,B) & a(B,A) & b(A,C) & b(C,A)

We thus know that in any CM, a(X,Y) maps to a(B,A). Thus, X->B and Y->A.

Then b(Y,Z) must map to b(A,C), and Z->C.

Thus, b(Z,W) -> b(C,B), and W->A

Now, a(W,X) -> a(A,B), and there are no more consequen-ces. We have a CM.