What We Know How to Do

Given sources that are described by views over global "EDB" predicates, we can:

a) If the query is a CQ over the global predicates, we can find a minimal set of CQ's over the views that produce all possible answers to the query obtainable from the sources.

    ❖ The LMSS theory and Information Manifold.

b) If the query is recursive Datalog, we can find a (possibly recursive) Datalog program that returns all possible answers to the query obtainable from the sources.

    ❖ Duschka/Genesereth, Infomaster.

Next Problem: Sources That Limit Access

- We can think of a view as answering a query.
- CQ views used so far can only answer the query "tell me all the tuples you have."
- Sometimes a source will not produce an entire view but will give you a piece of the view if you give it *bindings* for some of the arguments of the view.

Example

Amazon.com might be thought of as a title-author view.

- But in fact, it will not answer the query “tell me all the title-author pairs you have.”
- Rather, you have to give it an author or a title, and it will give you the associated titles or authors, respectively.

Views With Binding Patterns

We thus must consider views that have "binding patterns": certain arguments must be bound to a finite set of constants before we can use them. Two viewpoints:

1. First, we shall consider an IM-style framework, where we need to find a solution to a CQ query, but take into account the binding patterns of the views.
   - ✦ We can only solve the problem for equivalence, i.e., one solution that is equivalent to the query.
2. Next, we shall look into finding maximal, contained solutions to a query.
   - ✦ Even if the query is a CQ, the maximal solution may be recursive Datalog.

Adornments

To indicate binding patterns, we *adorn* a predicate or view with a string of b's and f's, indicating which of the arguments must be

- *Bound*, i.e., given a value in the query, and which are
- *Free* (supplied by the answer to the query).

Example

Consider a genealogy DB that can answer two query forms:

1. Given any individual C, find C's parents.

2. Find all the individuals who have parents specified by the information source.

If *parent* is the global "parent" we can express the source as supplying two adorned views:

$$v_1^{bf}(\mathtt{C,P}) \; \mathtt{:-} \; \mathtt{parent(C,P)}$$
$$v_2^{f}(\mathtt{C}) \; \mathtt{:-} \; \mathtt{parent(C,P)}$$

- That is, v_1 expects a binding for C, which becomes the first argument of *parent*, and it produces values for P, the second argument of *parent*.

- With no argument, v_2 produces the values of C that are a first argument of some *parent* fact.

Queries

Queries are likewise described by an adorned head that tells which arguments of the head are given constants as values.

- We do not use constants explicitly in the query.

Example

Here is a query for the grandparents of a particular individual C.

$answer_1^{bf}$`(C,G) :-`
```
        parent(C,P) & parent(P,G)
```

The query

$answer_2^{fb}$`(C,G) :-`
```
        parent(C,P) & parent(P,G)
```

looks similar, but asks for the grandchildren of an individual G.

Solutions

A solution S is again a CQ with views as subgoals, but now the views have adornments. To be a valid solution for query Q:

1. The adornments in the heads of S and Q must be the same.

2. $E(S) \subseteq Q$, ignoring adornments.

3. The adornments of every subgoal of S must be satisfied by the head of S and the subgoals to its left. That is, if X is a variable in a bound argument of subgoal j of S, then either:

   - X appears in a bound argument of the head of S, or

   - X appears in any argument of one of the first $j - 1$ subgoals of S.

Example

Consider our running example, with views v_1^{bf} and v_2^f and the query $answer_1^{bf}$ (find the grandparents of C). An appropriate solution is

$$
\begin{aligned}
&answer_1^{bf}\texttt{(C,G) :-}\\
&\qquad v_1^{bf}\texttt{(C,P) \& } v_1^{bf}\texttt{(P,G)}
\end{aligned}
$$

- This solution satisfies the binding-pattern condition because in the first subgoal, the first argument C is bound by the head, and in the second subgoal, the first argument P is bound by the first subgoal.

- The expansion of this solution is:

$$
\begin{aligned}
&answer_1^{bf}\texttt{(C,G) :-}\\
&\qquad \texttt{parent(C,P) \& parent(P,G)}
\end{aligned}
$$

- Since it is identical to the query, it is surely the unique minimal solution.

Another Example

Now consider the query $answer_2^{fb}$, where we are asked to find grandchildren.

- The following is *not* a valid solution:

$$answer_2^{fb}\texttt{(C,G) :-}$$
$$v_1^{bf}\texttt{(C,P) \& } v_1^{bf}\texttt{(P,G)}$$

- In the first subgoal, the first argument C does not receive a binding from the head, which only binds the second argument G.

  - Moreover, reversing the order of the subgoals does not help; the first argument P is then free.

Example (Continued): A Proper Solution

- However, using view v_2 we can produce a valid solution:

$$
\begin{aligned}
&answer_2^{fb}(\texttt{C,G}) \texttt{ :-} \\
&\qquad v_2^{f}(\texttt{C}) \ \texttt{\&} \\
&\qquad v_1^{bf}(\texttt{C,P}) \ \texttt{\&} \\
&\qquad v_1^{bf}(\texttt{P,G})
\end{aligned}
$$

- Now, the first subgoal, with predicate v_2, requires no bound argument but provides a binding for C.

- That binding allows the second subgoal to have its first argument bound, as v_1^{bf} requires.

- The third subgoal also has its required binding for the first argument.

- When we expand this solution, we get:

$answer_2^{fb}$`(C,G) :-`

```
        parent(C,X) &
        parent(C,P) &
        parent(P,G)
```

- This expansion is not identical to the query, but happens to be equivalent to it.
  - ❖ Intuitively, the $parent(C, X)$ subgoal is superfluous; it appears only because of the requirement about binding patterns.

Size of Solutions With Adornments

- Important observation: The LMSS theorem about solutions needing no more subgoals than the query does not hold any more.
  - ❖ Here, the query has two subgoals, but the minimal valid solution has three.

Theorem

For CQ-views with binding patterns, a minimal solution S that is *equivalent to* (not just "contained in") query Q cannot have more subgoals than the sum of the number of subgoals and variables of Q.

Proof

Let Q have n subgoals and m variables. Let μ be a containment mapping from Q to $E = E(S)$, and ν a containment mapping from E to Q.

- Group the variables of S such that $X \equiv Y$ iff $\nu(X) = \nu(Y)$.
    - Note: ν maps E, not S, to Q. However, if the view definitions are safe, all variables of S are also variables of E (think about it!).
- There cannot be more than m equivalence classes.
- Merge equivalent variables into one, to get a new solution S' that has no more than m variables.
- ν is also a containment mapping from $E' = E(S')$ to Q.

- We can modify μ to be a containment mapping from Q to E'. If $\mu(A) = B$, and B was replaced in S' and E' by equivalent variable C, then set $\mu'(A) = C$. Otherwise, $\mu'(A) = \mu(A)$.
    - ❖ Notice that this merging process cannot create a violation of the containment-mapping requirement.
- Now, consider μ'. If there is a subgoal of S' that neither:
    1. Generates in E' a subgoal that is in the target of μ', nor

2. Has the first occurrence of some variable in S', then we can eliminate this subgoal from S' to create S''.

- S'' is a shorter equivalent to Q. In proof:
    - ❖ Variables in S' that were bound at some point remain bound at that point in S''.
    - ❖ μ' and ν continue to serve as containment mappings.

Finding All Solutions With Adornments

General idea: obtain bindings for "domains," which we associate with the variables of a query.

- Important assumption: While, knowing that a view requires an integer first argument, we could ask the view about 1, about 2, about $\cdots$, we're not going to allow such query-answering strategies.
- Rather, we assume that any binding we use was either
  1. Obtained from the query, or
  2. Obtained from the answer to some other query and comes from the right "domain."

Example

The following example illustrates what is possible. We shall later explain where the solution came from.

- Global predicate: $ta(T, A) =$ there is a book with title T and author A.

- Source views:

  v_1^{bf}`(X,Y) :- ta(X,Y)`
  v_2^{fb}`(X,Y) :- ta(X,Y)`
  v_3^{f}`(Y) :- ta(X,Y)`

- Query: `ans(A) :- ta(T,A)`; i.e., find all the authors.

Solution Outline

1. Write rules that extract all possible *ta* facts, given bindings for zero or more of title and author.

2. Write rules that extract bindings for title and author from the *ta* facts.

Inversion of the *ta* Rules

$$\texttt{ta(X,Y) :- } v_1^{bf}\texttt{(X,Y)}$$
$$\texttt{ta(X,Y) :- } v_2^{fb}\texttt{(X,Y)}$$
$$\texttt{ta(g(Y),Y) :- } v_3^{f}\texttt{(Y)}$$

Rules for Domains

```
author(A) :- ta(T,A)
title(T) :- ta(T,A)
```

Rules for *ta* Facts

Note: the *ta* here is not the global *ta* concept, but the computable subset of that abstract predicate.

$\texttt{ta(T,A) :- title(T) \& } v_1^{bf}\texttt{(T,A)}$

$\texttt{ta(T,A) :- author(A) \& } v_2^{fb}\texttt{(T,A)}$

$\texttt{ta(g(A),A) :- } v_3^{f}\texttt{(A)}$

- Third rule for *ta* turns out to be useless except for starting off the set of *author* facts.

The Query

```
ans(A) :- ta(T,A)
```

Function-Symbol-Free Expansion of ans

(1) $\texttt{ans(A) :- } v_3^f\texttt{(A)}$

(2) $\texttt{ans(A) :- } v_3^f\texttt{(A) \& } v_2^{fb}\texttt{(T,A)}$

(3) $\texttt{ans(A) :- } v_3^f\texttt{(B) \& } v_2^{fb}\texttt{(T,B) \&}$
$v_1^{bf}\texttt{(T,A)}$

(4) $\texttt{ans(A) :- } v_3^f\texttt{(B) \& } v_2^{fb}\texttt{(T,B) \&}$
$v_1^{bf}\texttt{(T,A) \& } v_2^{fb}\texttt{(S,A)}$

(5) $\texttt{ans(A) :- } v_3^f\texttt{(B) \& } v_2^{fb}\texttt{(T,B) \&}$
$v_1^{bf}\texttt{(T,C) \& } v_2^{fb}\texttt{(S,C) \&}$
$v_1^{bf}\texttt{(S,A)}$

. . .

- Even-numbered expansions are contained in the previous odd-numbered expansion.
- But all the odd-numbered expansions are independent (no containments).

Algorithm

Suppose we have a set of global predicates, a set of views with binding patterns, defined in terms of the global predicates, and a query in terms of those global predicates.

1. Decide on the domains of the global predicates. That is, group the arguments into sets that will be assumed to use the same values.

2. For each domain D, create an IDB predicate $d(X)$, with the interpretation that X is a member of domain D.

3. If the ith argument of global predicate p belongs to domain D, then add the rule

   ```
   d(X) :- p(...X...)
   ```

   where the X appears in the ith argument of p, and all other arguments have new variables.

   - These rules extract new domain elements from discovered facts.

4. Invert the rules for the views.

5. For each inverted rule `p(X...) :- v(Y...)` that helps define a global predicate p, add a rule with:

   a) p in the head.

   b) The v-subgoal in the body.

   c) Additional subgoals of the form $d(X)$ if the v-subgoal has X in a bound argument position.

   ❖ These rules simulate calls to a view with properly chosen bindings for the bound arguments.

6. If necessary, use the Duschka/Genesereth procedure to replace the use of function symbols by new predicates that represent their use.

Example (Revisited)

1. The title and author attributes are assumed to belong to different domains.

2. We use *title* and *author* as the corresponding domain predicates.

3.

```
author(A) :- ta(T,A)
title(T) :- ta(T,A)
```

4.

$$\texttt{ta(T,A) :- } v_1^{bf}\texttt{(T,A)}$$
$$\texttt{ta(T,A) :- } v_2^{fb}\texttt{(T,A)}$$
$$\texttt{ta(g(A),A) :- } v_3^{f}\texttt{(A)}$$

5.

$$\texttt{ta(T,A) :- title(T) \& } v_1^{bf}\texttt{(T,A)}$$
$$\texttt{ta(T,A) :- author(A) \& } v_2^{fb}\texttt{(T,A)}$$
$$\texttt{ta(g(A),A) :- } v_3^{f}\texttt{(A)}$$

6.

$$
\begin{array}{l}
\texttt{author(A) :- ta(T,A)} \\
\texttt{author(A) :- } v_3^{f}\texttt{(A)} \\
\texttt{title(T) :- ta(T,A)} \\
\texttt{ta(T,A) :- title(T) \& } v_1^{bf}\texttt{(T,A)} \\
\texttt{ta(T,A) :- author(A) \& } v_2^{fb}\texttt{(T,A)}
\end{array}
$$