Getting All You Can Out of Views

- The situation is that we are given a collection of views and a query (possibly recursive).
  - We want to find all the answers to the query that we can using the views.
- This technology, due to Oliver Duschka, comes from "Infomaster," a project of Prof. Mike Genesereth.

Example

We have a parent EDB relation, with ancestors defined in the usual way:

```
anc(X,Y) :- par(X,Y)
anc(X,Y) :- par(X,Z) & anc(Z,Y)
```

but the only view we have tells about grandparents:

```
v1(X,Y) :- par(X,Z) & par(Z,Y)
```

If we want the most ancestor facts that we can obtain from the view v_1 (not using the EDB, which is only abstract in applications like IM), then we should use the program:

```
anc(X,Y) :- v1(X,Y)
anc(X,Y) :- v1(X,Z) & anc(Z,Y)
```

which gives us the even-distance ancestors that can be composed of facts in v_1, and nothing else.

Key Idea: Skolemization

1. Replace existential variables in the view definitions by new function symbols applied to the variables of the head.
   - The function symbols so used are called *Skolem functions*; it is a standard trick of logic to get rid of existential variables.
2. *Invert* the view definitions, so they are EDB predicates with function symbols defined in terms of a view.
   - Remember that the "EDB" predicates are really global, abstract concepts as in IM, not stored relations.

Example

To invert the view

```
v1(X,Y) :- par(X,Z) & par(Z,Y)
```

we get:

```
par(X,f(X,Y)) :- v1(X,Y)
par(f(X,Y),Y) :- v1(X,Y)
```

- Notice that `:-` is a misnomer as far as the view definition is concerned, but in the inverse rules it is simply an assertion that there are no bogus facts in the view.

A More Complex Example

Suppose we have EDB predicates (global concepts) $f(X, Y)$ and $m(X, Y)$, meaning that Y is the father or mother, respectively, of X.

- Our query is to find all "maternal ancestors" of an individual X, i.e., all females who are ancestors:

  r_1: `manc(X,Y) :- m(X,Y)`
  r_2: `manc(X,Y) :- f(X,Z) & manc(Z,Y)`
  r_3: `manc(X,Y) :- m(X,Z) & manc(Z,Y)`

- The available views are:

  v_1`(X,Y) :- f(X,Z) & m(Z,Y)`
  v_2`(X,Y) :- m(X,Y)`

Invert the Views

r_4: `f(X,g(X,Y)) :-` v_1`(X,Y)`
r_5: `m(g(X,Y),Y) :-` v_1`(X,Y)`
r_6: `m(X,Y) :-` v_2`(X,Y)`

Evaluating the Rules

In a sense, that's all there is to it. Treat all predicates except the views as IDB, and evaluate.

- Seminaive evaluation can produce tuples with function symbols, but these cannot be real answers to the query.
- Because all function symbols are in the heads of rules for "EDB" (global, conceptual) predicates, which have no other rules, we never introduce a function symbol within a function symbol, leading to a finite process.
- Thus, seminaive evaluation converges, and the set of *manc* facts without function symbols is the closest we can get to the true answer by using only the views.

Example of Inference

Suppose $v_1(a,b)$. Then we can infer:

- $m(g(a,b),b)$ by r_5.
- $manc(g(a,b),b)$ by r_1.
- $f(a,g(a,b))$ by r_4.
- $manc(a,b)$ by r_2.

Formal Elimination of Function Symbols

If you feel uncomfortable with function symbols floating around, there is a systematic way to rewrite the rules so there are no function symbols at all.

- Create new versions of the rules by using any pattern with function symbols that appears in a head, and using that pattern in subgoals with which the pattern can be unified.
- Then, invent a new predicate for each pattern of arguments in each IDB predicate.
  - ❖ The new predicate represents where the function symbols are found.
  - ❖ But the predicate itself has only variables as arguments, no function symbols.

Example

Continue with the *manc* rules.

- Initially, r_4 and r_5 have heads with function symbols.

  r_4: $f(.,g(.,.))$ is the pattern for rule r_4. Use it in r_2 to get:

  ```
  r7: manc(X,Y) :- f(X,g(A,B)) &
                   manc(g(A,B),Y)
  ```

  r_5: The head of r_5 has pattern $m(g(.,.),.)$, which unifies with the m subgoals in r_1 and r_3:

  ```
  r8: manc(g(C,D),Y) :- m(g(C,D),Y)
  r9: manc(g(C,D),Y) :- m(g(C,D),Z) &
                        manc(Z,Y)
  ```

- Now, r_8 and r_9 have new heads with function symbols; the patterns are both $manc(g(.,.),.)$.

- Thus we must use $manc(g(.,.),.)$ in r_2, r_3, r_7, and r_9.
  - Rules r_2 and r_7 yield nothing new.
  - r_3 and r_9 yield, respectively:

r_{10}: `manc(X,Y) :- m(X,g(E,F))`
`            & manc(g(E,F),Y)`
r_{11}: `manc(g(C,D),Y) :- m(g(C,D),g(E,F))`
`            & manc(g(E,F),Y)`

Cleaning Up the Rules

To get a program that computes the maximum answer from the views, we can:

1. Replace atoms with function symbols by equivalent, new predicates that have variables as arguments.
2. Substitute for the "EDB" (global) predicates in terms of the views, but only where no function symbols are introduced into the rules.
   - Justified because the views themselves have no data with function symbols, and other predicates have already had all possible forms with function symbols covered by other rules.

Example (Continued)

- Use $manc1(X,Y,Z)$ for $manc(g(X,Y),Z)$.
  - No other variants of $manc$ are needed in this simple example.

r_1: Only r_6 can be used for the m subgoal of r_1 (r_5 would introduce function symbols in the head, and we already generated r_8, an appropriate variant of r_1 where the m subgoal has been unified with the pattern of the head of r_5).

r_1: `manc(X,Y) :-` v_2`(X,Y)`

r_2: No suitable replacement for the f subgoal exists.

r_3: As for r_1, it is necessary only to use r_6, yielding:

r_3: `manc(X,Y) :-` v_2`(X,Z) & manc(Z,Y)`

r_4–r_6: These inversion rules will be eliminated.

---

r_7: We must:

a) Use r_4 for the f subgoal, which requires that A and X be unified because of the form of the head of r_4.

b) Use $manc1$ in place of $manc$ in the second subgoal.

r_7: `manc(X,Y) :-` v_1`(X,B) & manc1(X,B,Y)`

r_8: Use r_5 for the m subgoal; unification of D and Y is necessary.

r_8: `manc1(C,Y,Y) :-` v_1`(C,Y)`

r_9: Similar to r_8, but D unified with Z.

r_9: `manc1(C,Z,Y) :-` v_1`(C,Z) & manc(Z,Y)`

r_{10}: Neither r_5 nor r_6 allows unification with the m subgoal of r_{11} without introducing function symbols, so this rule cannot be used.

r_{11}: Same problem as r_{10}.

Summary of Rules

r_1: `manc(X,Y) :-` v_2`(X,Y)`
r_3: `manc(X,Y) :-` v_2`(X,Z) & manc(Z,Y)`
r_7: `manc(X,Y) :-` v_1`(X,B) & manc1(X,B,Y)`
r_8: `manc1(C,Y,Y) :-` v_1`(C,Y)`
r_9: `manc1(C,Z,Y) :-` v_1`(C,Z) & manc(Z,Y)`

- If we get rid of $manc1$ by expanding its subgoal in r_7 in both possible ways, we get:

  `manc(X,Y) :-` v_2`(X,Y)`
  `manc(X,Y) :-` v_2`(X,Z) & manc(Z,Y)`
  `manc(X,Y) :-` v_1`(X,B) & manc(B,Y)`
  `manc(X,Y) :-` v_1`(X,Y)`

- That's beginning to make sense; it says that we can concatenate either of the views in any possible way, since each represents a chain ending in a female.

- But note that it doesn't get all maternal ancestors, e.g., my Father's Father's Mother.