Using Views to Implement Datalog Programs

Inverse Rules

Duschka's Algorithm

Inverting Rules

- Idea: "invert" the view definitions to give the global predicates definitions in terms of views and function symbols.
- Plug the globals' definitions into the body of the query to get a direct expansion of the query into views.
- Even works when the query is a program.

Inverting Rules --- (2)

- But the query may have function symbols in its solution, and these symbols actually have no meaning.
- We therefore need to get rid of them.
- Trick comes from Huyn -> Qian -> Duschka.

Skolem Functions

- Logical trick for getting rid of existentially quantified variables.
- In terms of safe Datalog rules:
  - For each local (nondistinguished) variable *X*, pick a new function symbol *f* (the Skolem constant).
  - Replace *X* by *f* (head variables).

Example

```
v(X,Y) :- p(X,Z) & p(Z,Y)
```

- Replace Z by f(X,Y) to get:

```
v(X,Y) :- p(X,f(X,Y)) &
            p(f(X,Y),Y)
```

- Intuition: for v(X,Y) to be true, there must be some value, depending on X and Y, that makes the above body true.

HQD Rule Inversion

- Replace a Skolemized view definition by rules with:
  1. A subgoal as the head, and
  2. The view itself as the only subgoal of the body.

Example

```
v(X,Y) :- p(X,f(X,Y)) &
          p(f(X,Y),Y)
```

becomes:

```
p(X,f(X,Y)) :- v(X,Y)
p(f(X,Y),Y) :- v(X,Y)
```

Running Example: Maternal Ancestors

- Global predicates:
  - m(X,Y) = "*Y* is the mother of *X*."
  - f(X,Y) = "*Y* is the father of *X*."
- manc rules:

r1: `manc(X,Y) :- m(X,Y)`

r2: `manc(X,Y) :- f(X,Z) & manc(Z,Y)`

r3: `manc(X,Y) :- m(X,Z) & manc(Z,Y)`

Example --- Continued

- The views:

```
v1(X,Y) :- f(X,Z) & m(Z,Y)
v2(X,Y) :- m(X,Y)
```

- Inverse rules:

r4: `f(X,g(X,Y)) :- v1(X,Y)`

r5: `m(g(X,Y),Y) :- v1(X,Y)`

r6: `m(X,Y) :- v2(X,Y)`

Evaluating the Rules

- Treat views as EDB.
- Apply seminaïve evaluation to query (Datalog program).
- In general, function symbols -> no convergence.

Evaluating the Rules

- But here, all function symbols are in the heads of global predicates.
- These are like EDB as far as the query is concerned, so no nested function symbols occur.
- One level of function symbols assures convergence.

Example

```
r1: manc(X,Y) :- m(X,Y)
r2: manc(X,Y) :- f(X,Z) & manc(Z,Y)
r3: manc(X,Y) :- m(X,Z) & manc(Z,Y)
r4: f(X,g(X,Y)) :- v1(X,Y)
r5: m(g(X,Y),Y) :- v1(X,Y)
r6: m(X,Y) :- v2(X,Y)
```

- Assume v1(a,b).

Example --- (2)

r1: `manc(X,Y) :- m(X,Y)`

r2: `manc(X,Y) :- f(X,Z) & manc(Z,Y)`

r3: `manc(X,Y) :- m(X,Z) & manc(Z,Y)`

r4: `f(a,g(a,b)) :- v1(a,b)`

r5: `m(g(a,b),b) :- v1(a,b)`

r6: `m(X,Y) :- v2(X,Y)`

- Assume v1(a,b).

Example --- (3)

```
r1: manc(g(a,b),b) :- m(g(a,b),b)
r2: manc(X,Y) :- f(X,Z) & manc(Z,Y)
r3: manc(X,Y) :- m(X,Z) & manc(Z,Y)
r4: f(a,g(a,b)) :- v1(a,b)
r5: m(g(X,Y),Y) :- v1(X,Y)
r6: m(X,Y) :- v2(X,Y)
```

- Assume v1(a,b).

Example --- (4)

r1: `manc(X,Y) :- m(X,Y)`

r2: `manc(a,b) :- f(a,g(a,b)) &`
`                    manc(g(a,b),b)`

r3: `manc(X,Y) :- m(X,Z) & manc(Z,Y)`

r4: `f(X,g(X,Y)) :- v1(X,Y)`

r5: `m(g(X,Y),Y) :- v1(X,Y)`

r6: `m(X,Y) :- v2(X,Y)`

- Assume v1(a,b).

Example --- Concluded

- Notice that given v1(a,b), we were able to infer manc(a,b), even though we never found out what the value of g(a,b) [the father of *a*] is.

Rule-Rewriting

- Duschka's approach moves the function symbols out of the seminaïve evaluation and into a rule-rewriting step.
- In effect, the function symbols combine with the predicates.
  - Possible only because there are never any nested function symbols.

Necessary Technique: Unification

- We unify two atoms by finding the simplest substitution for the variables that makes them identical.
- Linear-time algorithm known.

Example

- The unification of p(f(X,Y), Z) and p(A,g(B,C)) is p(f(X,Y),g(B,C)).
- Uses A -> f(X,Y); Z -> g(B,C); identity mapping on other variables.
- p(X,X) and p(Y,f(Y)) have no unification.
- Neither do p(X) and q(X).

Elimination of Function Symbols

- Repeat:
  1. Take any rule with function symbol(s) in the head.
  2. Unify that head with any subgoals, of any rule, with which it unifies.
     - But first make head variables be new,unique symbols.
- Finally, replace IDB predicates + function-symbol patterns by new predicates.

Example

```
r1: manc(X,Y) :- m(X,Y)
r2: manc(X,Y) :- f(X,Z) & manc(Z,Y)
r3: manc(X,Y) :- m(X,Z) & manc(Z,Y)
r4: f(X,g(X,Y)) :- v1(X,Y)
r5: m(g(X,Y),Y) :- v1(X,Y)
r6: m(X,Y) :- v2(X,Y)
```

Unify

Example --- (2)

```
r2: manc(X,Y) :- f(X,Z) & manc(Z,Y)
r4: f(A,g(B,C)) :-
```

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

Important point: in the unification of *X* and *A*, any variable could be used, but it must appear in both these places.

Example --- (3)

```
r1: manc(X,Y) :- m(X,Y)
r3: manc(X,Y) :- m(X,Z) & manc(Z,Y)
r5: m(g(A,B),C) :-
```

```
r8: manc(g(A,B),Y) :- m(g(A,B),Y)
r9:manc(g(A,B),Y) :- m(g(A,B),Z) &
                             manc(Z,Y)
```

Example --- (4)

- Now we have a new pattern: `manc(g(A,B),C)`.
- We must unify it with `manc` subgoals in r2, r3, r7, and r9.
- r2 and r7 yield nothing new, but r3 and r9 do.

Example --- (5)

```
r3: manc(X,Y) :- m(X,Z) & manc(Z,Y)
r9: manc(g(A,B),Y) :- m(g(A,B),Z) &
                                 manc(Z,Y)
    manc(g(C,D),E)
--------------------------------------------
r10: manc(X,Y) :- m(X,g(A,B)) &
                         manc(g(A,B),Y)
r11: manc(g(A,B),Y) :-
          m(g(A,B),g(C,D))
          & manc(g(C,D),Y)
```

Cleaning Up the Rules

1. For each IDB predicate (`manc` in our example) introduce a new predicate for each function-symbol pattern.
2. Replace EDB predicates (*m* and *f* in our example) by their definition in terms of views, but only if no function symbols are introduced.

Justification for (2)

- A function symbol in a view is useless, since the source (stored) data has no tuples with function symbols.
- A function symbol in an IDB predicate has already been taken care of by expanding the rules using all function-symbol patterns we can construct.

New IDB Predicates

- In our example, we only need `manc1` to represent the pattern manc(g(.,.),.).
- That is: `manc1(X,Y,Z) = manc(g(X,Y),Z).`

Example --- r1

r1: `manc(X,Y) :- m(X,Y)`

r5: `m(g(X,Y),Y) :- v1(X,Y)`

r6: `m(X,Y) :- v2(X,Y)`

Substitution OK; yields `manc(X,Y) :- v2(X,Y)`

Illegal --- yields g in head. Note the case manc(g(.,.),.) is taken care of by r8.

Example --- r2 and r3

r2: `manc(X,Y) :- f(X,Z) & manc(Z,Y)`

r3: `manc(X,Y) :- m(X,Z) & manc(Z,Y)`

r4: `f(X,g(X,Y)) :- v1(X,Y)`

r5: `m(g(X,Y),Y) :- v1(X,Y)`

r6: `m(X,Y) :- v2(X,Y)`

Illegal --- put function symbol g in manc.

OK; yields

```
manc(X,Y) :- v2(X,Z) &
             manc(Z,Y)
```

Example --- r4, r5, r6

- The inverse rules have played their role and do not appear in the final rules.

Example --- r7

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

r4: `f(X,g(X,Y)) :- v1(X,Y)`

Unify. Note no function symbols are introduced, but $X = B$.

Replace by `manc1(B,C,Y)`.

r7: `manc(X,Y) :- v1(X,C) &`
`manc1(X,C,Y)`

Example --- r8 and r9

r8: `manc(g(A,B),Y) :- m(g(A,B),Y)`

r9: `manc(g(A,B),Y) :- m(g(A,B),Z) & manc(Z,Y)`

r5: `m(g(X,Y),Y) :- v1(X,Y)`

Become `manc1(A,B,Y)`

Unify; set $B = Y$.

Unify; set $B = Z$.

r8: `manc1(A,Y,Y) :- v1(A,Y)`

r9: `manc1(A,Z,Y) :- v1(A,Z)& manc(Z,Y)`

Example --- r10 and r11

- No substitutions possible --- unifying *m* –subgoal with head of r5 or r6 introduces a function symbol into the view subgoal.

Summary of Rules

```
r1: manc(X,Y) :- v2(X,Y)
r3: manc(X,Y) :- v2(X,Z) & manc(Z,Y)
r7: manc(X,Y) :- v1(X,C) &
                 manc1(X,C,Y)
r8: manc1(A,Y,Y) :- v1(A,Y)
r9: manc1(A,Z,Y) :- v1(A,Z)&
                 manc(Z,Y)
```

Finishing Touch: Replace `manc1`

Substitute the bodies of r8 and r9 for the `manc1` subgoal of r7.

r7: `manc(X,Y) :- v1(X,C) &`
`manc1(X,C,Y)`

r8: `manc1(A,Y,Y) :- v1(A,Y)`

r9: `manc1(A,Z,Y) :- v1(A,Z)&`
`manc(Z,Y)`

Becomes another `v1(X,C)`.

Becomes `v1(X,C) & manc(C,Y)`

Final Rules

```
r1:   manc(X,Y) :- v2(X,Y)
r3:   manc(X,Y) :- v2(X,Z) &
                      manc(Z,Y)
r7-8: manc(X,Y) :- v1(X,Y)
r7-9: manc(X,Y) :- v1(X,C) &
                      manc(C,Y)
```