The Bucket Algorithm

- We can "answer queries using views" by trying all CQ's with no more (view) subgoals than the query has subgoals.
- However, a more organized exploration of the possibilities, called the *bucket algorithm* looks at how views can "cover" each of the query subgoals and each variable of the query.
- Basic idea due to Levy; improved to consider the query variables as well as the subgoals by Prasenjit Mitra (and independently by Levy and Pottinger).

Limits on the Containment Mapping From Query to Expansion

The key to the bucket algorithm is to understand what can happen to variables in the query, when they are mapped to variables in the expansion.

- Each subgoal of the query must be *covered* by some view in the solution. That is, the query subgoal must map to some subgoal in some expansion, or the expansion is not contained in the query.
- We must differentiate between *distinguished variables* (those that appear in the query head or the head of a view definition) and nondistinguished variables (others).
- We must also differentiate between *unique* nondistinguished variables (those that appear only once in the query) and *shared* variables (which appear more than once).

R1. A distinguished variable in the query must map to a distinguished variable (of some view) in the expansion.

R2. A shared variable X must either map to a distinguished variable of a view expansion or all query subgoals involving X must map to the same nondistinguished variable in the expansion of a single view.

  - The reason is that two view expansions can never have the same nondistinguished variable.
  - Even if the local variable name is the same in the view definition, the expansions must use different names.

Examples

Figure 0.1 shows one consequence of rules $R1$ and $R2$.

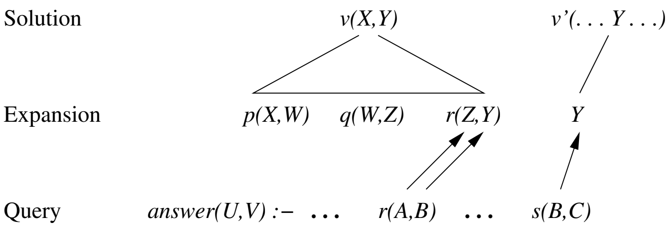

Figure 0.1: Covering the query subgoal $r(A, B)$

- A is a unique, nondistinguished variable. It can map to the nondistinguished variable Z in the expansion of view v.
- B is a shared, nondistinguished variable. One possible treatment, shown in Fig. 0.1, is that B maps to a distinguished variable, Y, and other occurrences of B, suggested by the subgoal $s(B, C)$, are mapped so that occurrence of B maps to a distinguished variable of some other view.
  - Note that in designing the solution, we have the freedom to equate the variables of the views, and thus to use the same distinguished variable in two or more view expansions.
  - This capability makes it possible to map the occurrences of B to the same variable, coming from expansions of different views.

Figure 0.2 suggests what happens if a shared variable is forced to map to a nondistinguished variable in a view expansion. The s subgoal now has noplace to go, since there is no s subgoal in the expansion of v, and another expansion could not have Z as a variable.

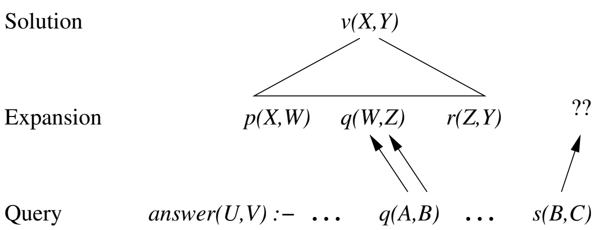

Figure 0.2: We cannot map B to local variable Z

On the other hand, if the other subgoal(s) with B can also map to subgoals in the same expansion, then B can be "covered." Figure 0.3 shows what happens when the s subgoal becomes and r subgoal.

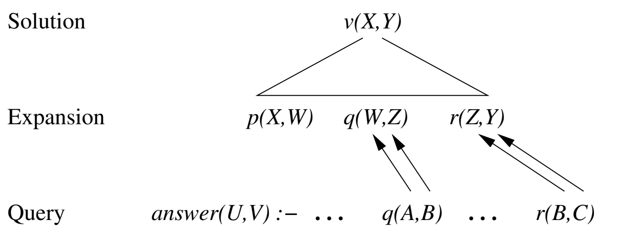

Figure 0.3: Now, both subgoals with B can map to subgoals in the same view expansion

Now, consider what happens when we have to map a distinguished variable of the query, such as A in Fig. 0.4. It is essential that A map to a distinguished variable such as X of a view expansion, because that is the only way the target variable can appear in the head of the solution.

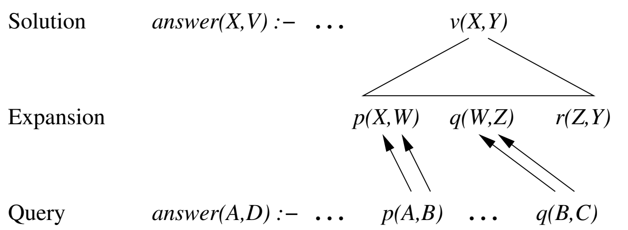

Figure 0.4: Handling a distinguished variable

Buckets for Subgoals and Variables

Each shared variable and each subgoal gets a "bucket" of views (and one or more subgoals within that view's expansion) whose expansion can cover that subgoal or the set of subgoals that have the given variable. Here are the bucket-construction rules:

1. If $p(A_1, \ldots, A_n)$ is a subgoal, then its bucket includes every view v, and every p-subgoal $p(X_1, \ldots, X_n)$ in the expansion of v such that:

   (a) There is a containment mapping from $A_1, \ldots, A_n$ to $X_1, \ldots, X_n$. (Note: the only reason there might not be is if there were duplicate occurrences of variables among the A's.)

   (b) If A_i is a distinguished variable of the query, then X_i is a distinguished variable of the definition of v.

   (c) If A_i is a shared variable, then X_i is a distinguished variable of the definition of v.

2. If B is a shared variable, then the bucket for B includes each view v and each set of subgoals in the expansion of v such that there is a containment mapping from all the query subgoals containing B to that subset of the subgoals in v's expansion.

   - The containment mapping must map distinguished (of the query) to distinguished (of the view definition).

Example

Consider the following views v and w, and query:

```
v(X,Y) :- p(X,Z) & p(Z,Y)
w(U,V) :- p(U,S) & p(S,T) & p(T,V)

query(A,B) :- p(A,C) & p(C,D) & p(D,E) &
              p(E,F) & p(F,G) & p(G,B)
```

Think of p as "parent," v as "grandparent," and w as "great-grandparent."

- The bucket for $p(A,C)$ is empty, because A is distinguished and C is shared. But no view subgoal has two distinguished variables.

- The bucket for $p(C,D)$ is empty because both variables are shared, and again no view subgoal has two distinguished variables.

- Similarly, the buckets for the other query subgoals are empty.

- The bucket for shared variable C includes $\{p(X,Z),\ p(Z,Y)\}$ from v, and $\{p(U,S),\ p(S,T)\}$ from w.

  – It does not include $\{p(S,T),\ p(T,V)\}$ from w, because $p(A,C)$ would have to map to $p(S,T)$, and A is distinguished, but S is not.

- The bucket for shared variable D includes $\{p(X,Z),\ p(Z,Y)\}$ from v, and both $\{p(U,S),\ p(S,T)\}$ and $\{p(S,T),\ p(T,V)\}$ from w.

  – Note that we cannot at this point guarantee a consistent treatment of the variables C and E that are involved in the mapping of the subgoals with D; a more careful analysis can check that the requirements of these shared variables can be met in conjuction with D's requirements.

We have to put together some views whose expansions cover all the subgoals.

1. One possibility is to use expansions of v as the buckets for shared variables C, E, and G.

   - The other shared variables fortunately cna map to distinguished variables of the expansions, so we can handle that sharing by equating the variables in the view subgoals of the solution.
   - Likewise, the distinguished variables of the query can map to distinguished variables of the expansions.
   - The resulting solution:

     ```
     query(A,B) :- v(A,H) & v(H,I) & v(I,B)
     ```

2. Another solution uses one expansion of w to cover the buckets for both C and D and a second expansion of w for F and G.

   - Again, the sharing of E can be handled by using the same variable in the two subgoals of the solution, and the distinguished variables of the query fortuitously map to distinguished variables of the view expansions.
   - The solution:

     ```
     query(A,B) :- w(A,H) & w(H,B)
     ```

3. Notice there is no covering with one use of v and one of w. However, we do it, one query subgoal needs to be covered by a third view expansion, and there is no way to make both variables of the "lost" subgoal map to distinguished variables of that expansion.