Using CQ Theory in Information Integration

Yes; this stuff really does get used in systems. We shall talk about three somewhat different systems that use the theory in various ways:

1. *Information Manifold*, developed by Alon Levy at ATT Research Labs (Levy is now at U. Washington).
2. *Infomaster*, developed at Stanford by Mike Genesereth and his group.
3. *Tsimmis*, developed in the Stanford DB group.

Two Broad Approaches

1. *View Centric*: There is a set of global predicates. Information sources are described by what they produce, in terms of the global predicates.
   - *View* = query describing what a source produces.
   - Global predicates behave like EDB, even though they are not stored and don't really exist.
   - Queries in terms of the global predicates are answered by piecing together views.
2. *Query-Centric*: A *mediator* exports global predicates.
   - Queries about these global predicates are translated by the mediator into queries at the sources and the answer is pieced together from the source responses.
   - Source predicates play the role of EDB.
   - Predicates exported by the mediator are defined by "views" of the source predicates.

Building Queries From Views

Information Manifold (IM) is built on the principle that there is a global set of predicates, and information sources are described in terms of what they can say about those predicates.

- We describe each information source by a set of *views* that they can provide.
  - Views are expressed as CQ's whose subgoals use the global predicates.

- Queries are also CQ's about the global predicates.

Fundamental Question:

Given a query and a collection of views, how do we find an expression *using the views only*, that is equivalent to the query.

- Remember: equivalence = containment in both directions.
- Sometimes equivalence is not possible; we need to find a query about the views that is maximally contained in the query.
- In IM, we really want all CQ's whose subgoals are views and that are contained in the query, since each expression may contribute answers to the query.
  - ❖ Exception: if one CQ is contained in another, then we don't need the contained CQ.

Example

Let us consider an integrated information system about employees of a company.

- Global predicates:

  $emp(E)$ = E is an employee
  $phone(E, P)$ = P is E's phone
  $office(E, O)$ = O is E's office
  $mgr(E, M)$ = M is E's manager
  $dept(E, D)$ = D is E's department

We suppose three sources, each providing one view:

```
v1(E,P,M) :- emp(E) & phone(E,P)
        & mgr(E,M)
v2(E,O,D) :- emp(E) & office(E,O)
        & dept(E,D)
v3(E,P) :- emp(E) & phone(E,P)
        & dept(E,toy)
```

1. View v_1, gives information about employees, their phones and managers.
2. View v_2 and gives information about the offices and departments of employees.
3. View v_3 provides the phones of employees, but only for employees in the Toy Department.

Interpretation of View Definitions

- A view definition gives properties that the tuples produced by the view must have.
- The view definition is *not* a guarantee that all such tuples are provided by the view.
- There is not even a guarantee that results produced by the two views are consistent.
  - E.g., there is no reason to believe the phone information provided by v_1 and v_3 is consistent.

Example

The constraint department = "Toy" is enforced by the subgoal $dept(E, toy)$ in the definition of v_3.

- This constraint would be important if we asked a query about employees known not to be in the Toy Department; we would not include v_3 in any solution.

Consider the query: "what are Sally's phone and office?" In terms of the global predicates:

```
q1(P,O) :- phone(sally,P) &
           office(sally,O)
```

- There are two *minimal* solutions to this query.
  - "Minimal" = not contained in any other solution that is also contained in the query.

```
a1(P,O) :- v1(sally,P,M) & v2(sally,O,D)
a2(P,O) :- v3(sally,P) & v2(sally,O,D)
```

If we expand the views in the rules for the answer, we get:

```
a1(P,O) :- emp(sally) & phone(sally,P)
    & mgr(sally,M) & emp(sally)
    & office(sally,O) & dept(sally,D)
a2(P,O) :- emp(sally) & phone(sally,P)
    & dept(sally,toy) & emp(sally)
    & office(sally,O) & dept(sally,D)
```

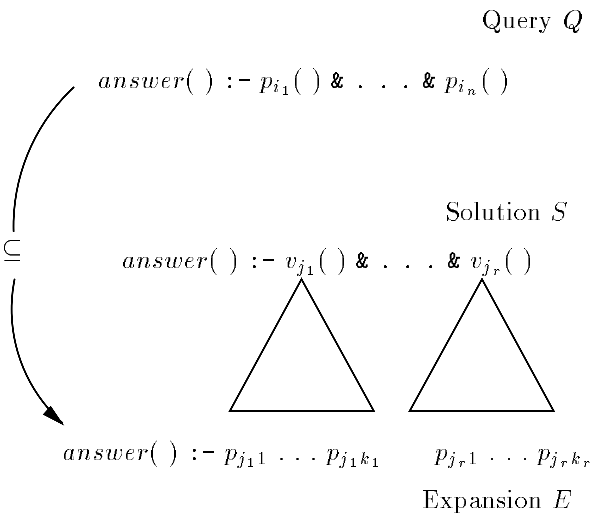
- Note these CQ's are not equivalent to q_1; they are the CQ's that come closest to q_1 while still being contained in q_1 and constructable from the views.

Selecting Solutions to a Query

The search for solutions by IM is based on a theorem that limits the set of CQ's that can possibly be useful.

- The search is exponential in principle but appears manageable in practice.

The Query-Expansion Process



Explanation of Expansion Diagram

- A query Q is given; solutions S are proposed, and each solution is *expanded* to a CQ $E = E(S)$ by replacing the view-subgoals in S by their definitions in terms of the global predicates.
  - ❖ As always, when replacing a subgoal by the body of a rule, be sure to use unique variables for the local variables in the rule body.
- A solution S is *valid* for Q if $E(S) \subseteq Q$.
- In principle, there can be an infinite number of valid solutions for a query Q.
  - ❖ Just add irrelevant subgoals to S; they may make the solution smaller, but it will still be contained in Q.
- Thus, we want only *minimal* solutions, those not contained in any other solution.

Important Reminder

Minimality is at the level of solutions, not expansions.

- Since views may provide different subsets of the global predicates, comparing expansions for containment *might* lead to false conclusions based on the (false) assumption that two views provided the same data.

Example

- Views:
  v_1`(X,Y) :- par(X,Y)`
  v_2`(X,Y) :- par(X,Y)`

- Query:

  `ans(X,Y) :- par(X,Y)`

- Solutions:
  `ans(X,Y) :-` v_1`(X,Y)`
  `ans(X,Y) :-` v_2`(X,Y)`

- The *expansions* of the solutions are each contained in the query, so they are valid solutions, and should be included.
  - ❖ They are in fact equivalent to the query, but that is irrelevant, since the "`:-`" in the view definitions is a misnomer; the views need not have *every* `par` fact.
- The solutions themselves (without expansion) are not contained in one another. Thus, neither can eliminate the other in the set of solutions.

Theorem

If S is a solution for query Q, and S has more subgoals than Q, then S is not minimal.

Proof

Look at the containment mapping from Q to $E(S)$.

- If S has more subgoals than Q, then there must be some subgoal g of S such that no subgoal of Q is mapped to any subgoal of $E(S)$ that comes from the expansion of g.
- If we delete g from S to make a new solution S', then $E(S') \subseteq Q$.
  - ❖ Proof: The containment mapping from Q to $E(S)$ is also a containment mapping from Q to $E(S')$.

- Moreover, $S \subseteq S'$.
  - Proof: The identity mapping on subgoals gives us the containment mapping.
  - Note this test must be carried out without expansion.
- Thus, S' is a valid solution that contains S in raw form (without expansion).

Example

Continuing the "employees" example, query q_1:

```
q1(P,O) :- phone(sally,P) &
           office(sally,O)
```

has two subgoals. Answers a_1 and a_2 each have two subgoals, so they might be minimal (they are!).

- However, the following answer:

  ```
  a3(P,O) :- v1(sally,P,M)
      & v2(sally,O,D) & v3(E,P)
  ```

  cannot be minimal, because it has three subgoals, more than q_1 does.
  - Note that a_3 is a_1 with the additional condition that Sally's phone must be the phone of somebody in the Toy Dept.
  - Thus, $a_3 \subseteq a_1$ without expansion, and a_3 cannot be minimal.

- The expansion of a_3 is:

```
a3(P,O) :- emp(sally) & phone(sally,P)
    & mgr(sally,M) & emp(sally)
    & office(sally,O) & dept(sally,D)
    & emp(E) & phone(E,P)
    & dept(E,toy)
```

  - Thus, $E(a_3) \subseteq q_1$, and a_3 is valid, although not minimal.