Refer

- A system developed by Walter Sujansky for transporting queries from one medical DB to another.
  - ❖ Focuses on relational DB's as the sources.
- Global model described by entities and relationships.
- Queries described in a Datalog-ish language with classes (entity sets) made explicit.
- Translation from the source relations to global entity sets and relationships written in relational algebra for each source.

Example

Our example will be all in Datalog and follows the "Sally's phone and office" pattern.

- The model will include entity sets Emp (which includes managers), Phone, Office, and Dept and relationships PhoneOf, OfficeOf, DeptOf, and MgrOf, from Emp to Phone, Office, Dept, and Emp, respectively.
  - ❖ We do not treat managers as a separate entity set, but we could.
- The particular source we shall use has two relations: $e_1(E, P, M)$ and $e_2(E, O, D)$, respectively giving for an employee E their phone and manager, and their office and department.

- For each of the entity sets and relationships we need to write rules (expressions of relational algebra in the real Refer) that define them in terms of the source relations.

  - For the entity sets:

```
emp(E) :- e1(E,P,M)
emp(E) :- e2(E,O,D)
phone(P) :- e1(E,P,M)
office(O) :- e2(E,O,D)
dept(D) :- e2(E,O,D)
```

  - And for the relationships:

```
phoneOf(E,P) :- e1(E,P,M)
mgrOf(E,M) :- e1(E,P,M)
officeOf(E,O) :- e2(E,O,D)
deptOf(E,D) :- e2(E,O,D)
```

* Now, consider the "usual" query about Sally's phone and office, but expressed in the Refer style.

    ❖ The query has subgoals that restrict "objects" (i.e., variables) to belong to the proper entity set, as well as subgoals representing relationships.

Q: `ans(P,O) :- emp(sally) &`
```
phone(P) & office(O) &
phoneOf(sally,P) & officeOf(sally,O)
```

- In order to ask query Q at the sources, Refer replaces each query subgoal by its definition. The solution thus consists of two rules (because entity set *emp*'s definition comes from two rules).

```
ans(P,O) :- e1(sally,P1,M1) &
    e1(E2,P,M2) & e2(E3,O,D3) &
    e1(sally,P,M4) & e2(sally,O,D5)

ans(P,O) :- e2(sally,O1,D1) &
    e1(E2,P,M2) & e2(E3,O,D3) &
    e1(sally,P,M4) & e2(sally,O,D5)
```

- Remember we must be careful to use new variables when a variable in a rule appears only in the body.

  - For instance, in the third subgoal of either rule, we have $e_2(E_3, O, D_3)$. It is the body of the rule

    ```
    office(O) :- e2(E,O,D)
    ```

    We have chosen E_3 and D_3 to replace E and D of that rule's body.

Minimizing CQ's

Refer forces us to tackle squarely an issue that has come up several times in an ad-hoc way: minimizing CQ's that have redundant subgoals. An important observation:

- Every CQ Q has a minimal equivalent CQ that consists of Q's head and a subset of its subgoals.
  - I.e., the *only* thing we need to do to find a minimal equivalent is throw away some subgoals.

Proof

Let Q be a CQ and P a minimal equivalent. Then there are containment mappings μ: $Q \rightarrow P$ and ν: $P \rightarrow Q$.

- Let Q' be the head of Q plus those subgoals of Q that are the target of subgoals of P under ν.

- $Q' \subseteq P$; ν proves it.

- $P \subseteq Q'$; μ proves it.

- Q' is a minimal equivalent to Q, because it cannot have more subgoals that P (because each subgoal of Q' is the target of one subgoal of P.

Minimizing Unions of CQ's

The same idea holds. Each union has a minimal equivalent union that is a subset of its CQ's.

- The selected CQ's in the union can in turn have their bodies minimized as above.

Example

Consider our running example. To help minimize the CQ's in the solution, remember:

1. The variables in the head (often called *distinguished variables*) must not change.

2. A variable that appears only once can be mapped to anything and presents no constraint at all on a containment mapping.

3. Variables can map to constants, but constants cannot map to variables.

   ❖ For example, in the first CQ, we can map $e_1(E_2, P, M_2)$ to $e_1(sally, P, M4)$ and thus eliminate the former subgoal.

- After eliminating the first three subgoals, we are left with minimal equivalent for the first CQ:

```
ans(P,O) :-
    e1(sally,P,M4) & e2(sally,O,D5)
```

- When we minimize the second CQ, we get exactly the same CQ.
- Thus, when we minimize the union, either can be eliminated, leaving the CQ above as the optimized answer.