MASSACHVSETTS INSTITVTE OF TECHNOLOGY
Department of Electrical Engineering and Computer Science
6.001—Structure and Interpretation of Computer Programs
Fall 2007

Recitation 21 — 11/21/2007 Solutions
Logic Programming: Query Language

Query Language

The evaluator for the query language, `qeval`, has different operations than the evaluators we've seen before. There are two main types of operations: assertions, and queries:

- Assertions add statements to the database of statements known to be true. Assertions can either be simple assertions such as:
  `(assert! (address (Bitdiddle Ben) (Slumerville (Ridge Road) 10)))`

  Or they can be compound `rule` statements which has the form:

  `(assert! (rule` ⟨*concl*⟩ ⟨*query*⟩`))`

  Which means, if the query matches, then add a new assertion which consists of *concl* with any variables filled in. Rules can apply recursively (though we'll later go into examples of where this can lead to problems).

  The simple assertions in the SICP example are on the last page.

- Queries

  Queries also come in simple and compound forms. A simple query with no variables such as `(salary (Bitdiddle Ben) 60000)` would either find that record if that statement is in the database, or not if it isn't.

  Variables can match any value, so `(salary ?x ?y)` would return a list of all the salary records for all employees.

  Compound queries such as
  `(and` ⟨*query*1⟩ ⟨*query*2⟩`)`
  `(or` ⟨*query*1⟩ ⟨*query*2⟩`)`
  find the intersection or union of the corresponding simple queries.

  `not` acts as a filter, removing any records where a query matches.

  A final special form, `lisp-value` takes a predicate and applies it to a set of values, for example, all employees with salary between 30000 and 50000 would be:

```
(and (salary ?p ?x)
     (lisp-value >= ?x 30000)
     (lisp-value <= ?x 50000))
```

Problems

1. Define a rule that says that person 1 can replace person 2 if either person 1 does the same job as person 2 or someone who does person 1's job can also do person 2's job, and if person 1 and person 2 are not the same person. (ex. 4.57 from SICP)

   ```
   (rule (can-replace ?p1 ?p2)
         (and (or
                (and (job ?p1 ?job)
                     (job ?p2 ?job))
                (and (job ?p1 ?job1)
                     (job ?p2 ?job2)
                     (can-do-job ?job1 ?job2)))
              (not (same ?p1 ?p2))))
   ```

   Using your rule, give queries that find the following:

   (a) All people who can replace Cy D. Fect;

   ```
   (can-replace (Fect Cy D) ?x)
   ```

   (b) All people who can replace someone who is being paid more than they are.

   ```
   (and (can-replace ?p1 ?p2)
        (salary ?p1 ?s1)
        (salary ?p2 ?s2)
        (lisp-value > ?s1 ?s2))
   ```

2. Define a rule that says that a person is a "big shot" in a division if the person works in the division but does not have a supervisor who works in the division. (ex 4.58 from SICP)

   ```
   (rule (big-shot ?p)
         (and (job ?p (?dept ?x))
              (supervisor ?p ?super)
              (job ?super (?superdept ?y))
              (not (same ?dept ?superdept))))
   ```

Infinite Loops

3. Consider the following definitions.

```
(assert! (married Minnie Mickey))
(married Mickey ?who)
(assert! (rule (married ?x ?y)
               (married ?y ?x)))
(married Mickey ?who)
```

The first call to `married` will return nothing. The second will go into an infinite loop. Why?

4. One of the following definitions for `outranked-by` works, and the other goes into an infinite loop. Find the difference and explain why:

```
(rule (outranked-by ?staff-person ?boss)
      (or (supervisor ?staff-person ?boss)
          (and (supervisor ?staff-person ?middle-manager)
               (outranked-by ?middle-manager ?boss))))

(rule (outranked-by ?staff-person ?boss)
      (or (supervisor ?staff-person ?boss)
          (and (outranked-by ?middle-manager ?boss)
               (supervisor ?staff-person ?middle-manager))))
```

Example assertions

These are the examples from Section 4.4.1 of SICP. To use these in the `qeval` evaluator, each of these would be wrapped with inside a call to `assert!`, but those are not shown

```
(address (Bitdiddle Ben) (Slumerville (Ridge Road) 10))
(job (Bitdiddle Ben) (computer wizard))
(salary (Bitdiddle Ben) 60000)
(address (Hacker Alyssa P) (Cambridge (Mass Ave) 78))
(job (Hacker Alyssa P) (computer programmer))
(salary (Hacker Alyssa P) 40000)
(supervisor (Hacker Alyssa P) (Bitdiddle Ben))
(address (Fect Cy D) (Cambridge (Ames Street) 3))
(job (Fect Cy D) (computer programmer))
(salary (Fect Cy D) 35000)
(supervisor (Fect Cy D) (Bitdiddle Ben))
(address (Tweakit Lem E) (Boston (Bay State Road) 22))
(job (Tweakit Lem E) (computer technician))
(salary (Tweakit Lem E) 25000)
(supervisor (Tweakit Lem E) (Bitdiddle Ben))
(address (Hacker Alyssa P) (Cambridge (Mass Ave) 78))
(job (Hacker Alyssa P) (computer programmer))
(salary (Hacker Alyssa P) 40000)
(supervisor (Hacker Alyssa P) (Bitdiddle Ben))
(address (Fect Cy D) (Cambridge (Ames Street) 3))
(job (Fect Cy D) (computer programmer))
(salary (Fect Cy D) 35000)
(supervisor (Fect Cy D) (Bitdiddle Ben))
(address (Tweakit Lem E) (Boston (Bay State Road) 22))
(job (Tweakit Lem E) (computer technician))
(salary (Tweakit Lem E) 25000)
(supervisor (Tweakit Lem E) (Bitdiddle Ben))
(address (Reasoner Louis) (Slumerville (Pine Tree Road) 80))
(job (Reasoner Louis) (computer programmer trainee))
(salary (Reasoner Louis) 30000)
(supervisor (Reasoner Louis) (Hacker Alyssa P))
(supervisor (Bitdiddle Ben) (Warbucks Oliver))
(address (Warbucks Oliver) (Swellesley (Top Heap Road)))
(job (Warbucks Oliver) (administration big wheel))
(salary (Warbucks Oliver) 150000)
(address (Scrooge Eben) (Weston (Shady Lane) 10))
(job (Scrooge Eben) (accounting chief accountant))
(salary (Scrooge Eben) 75000)
(supervisor (Scrooge Eben) (Warbucks Oliver))
(address (Cratchet Robert) (Allston (N Harvard Street) 16))
(job (Cratchet Robert) (accounting scrivener))
(salary (Cratchet Robert) 18000)
(supervisor (Cratchet Robert) (Scrooge Eben))
(address (Aull DeWitt) (Slumerville (Onion Square) 5))
(job (Aull DeWitt) (administration secretary))
(salary (Aull DeWitt) 25000)
(supervisor (Aull DeWitt) (Warbucks Oliver))
(can-do-job (computer wizard) (computer programmer))
(can-do-job (computer wizard) (computer technician))
(can-do-job (computer programmer)
            (computer programmer trainee))
(can-do-job (administration secretary)
            (administration big wheel))
```