

Topic: Logic programming

Lectures: Monday August 11, Tuesday August 12

Reading: Abelson & Sussman, Section 4.4.1–3

This assignment gives you practice writing logic programs. It's very much "above the line" since we don't expect you to know how the query system works. This homework is due at **midnight on Wednesday**, **August 10**. Please put your solutions into a file hw8-1.scm and submit electronically. Make sure to include your test cases, too.

```
To add an assertion: (assert! < conclusion>)

To add a rule: (assert! (rule < conclusion> < body (optional)>))

Anything else is a query.
```

The query interpreter is in the file ~cs61a/lib/query.scm. To initialize the interpreter type (query); to re-enter the main loop without reinitializing, type (query-driver-loop). Nothing—not even the rules for the same and append-to-form relations—is there when the interpreter is initialized.

Question 1. Do Exercise 4.56 in SICP. To load the database, type the following after loading query.scm:

```
STk> (initialize-data-base microshaft-data-base)
STk> (query-driver-loop)
```

The pattern (?a . ?b) matches any pair, so you can use it to print everything that is in the database.

Question 2. This question explores the unary arithmetic system described in lecture where numbers are represented as lists.

A. Note that summing two of these unary numbers merely involves joining the lists that represent them. We can define a rule for adding query numbers using append-to-form (Page 451):

```
;;; Query input:
  (assert! (rule (?a + ?b = ?c) (append-to-form ?a ?b ?c)))
Assertion added to data base.
;;; Query input:
  ((a a a a) + (a a a) = ?what)
;;; Query results:
  ((a a a a) + (a a a) = (a a a a a a a)) ;; 4 + 3 = 7

Devise rules to allow multiplication of query numbers:
;;; Query input:
  ((a a a a) * (a a a) = ?what)
;;; Query results:
  ((a a a a) * (a a a) = (a a a a a a a a a a a a)) ;; 4 * 3 = 12
```

The question continues on the next page.

B. Using your multiplication rule from above, implement a factorial relation for query numbers:

Question 3. We can interpret the query interpreter's failure to return any results as saying, "Your query was not consistent with any assertions I know or any rules I can apply on the basis of those assertions." This can be used to implement true/false queries where the interpreter echoes the query if it is true and displays no results otherwise. For example:

```
;;; Query input:
(deep-member a ((a) b))
;;; Query output:
(deep-member a ((a) b))
                              ;; true
;;; Query input:
(deep-member c ((b ((a)))))
;;; Query output:
                                    ;; false
;;; Query input:
(deep-member c ((b ((a c)))))
;;; Query output:
(deep-member c ((b ((a c)))))
;;; Query input:
(deep-member ?anything ())
;;; Query output:
                                 ;; false
```

Write rules for the deep-member relation that behaves as above.

Question 4. Write query rules for the assoc relation. It should work like this: