Assignment 4   C311   Fall 2005
Representation Independent Interpreters /
Logic Programming I

Revised 22 September
Due Sunday, September 25 at 6 p.m.

• Make sure you have read chapters one and nine of The Reasoned Schemer thoroughly before beginning Part II of this assignment.

• This is a long assignment. Get started early!

• Place your code in a file named a4.ss and submit it to Vincent.

Part I: Interpreters

Write the following four interpreters:

1. An interpreter value-of/proc/proc that uses procedural representation for closures and procedural representation for environments.

2. An interpreter value-of/proc/ds that uses procedural representation for closures and data structure representation for environments.

3. An interpreter value-of/ds/proc that uses data structure representation for closures and procedural representation for environments.

4. An interpreter value-of/ds/ds that uses data structure representation for closures and data structure representation for environments.
For each interpreter, make sure you define the five following helper procedures: `base-env`, `extend-env`, `apply-env`, `make-closure`, and `apply-closure`. To make sure all of your helper functions have unique names, suffix the helpers for a given interpreter with the same suffix as the interpreter (for example, `value-of/proc/proc` has `base-env/proc/proc`, `extend-env/proc/proc`, etc).

Make sure you include the `expressed-value→scheme-value` helper procedure in your definition.

Insert the following code into your file. When it loads, the result of calling `test-value-of` should confirm that all four interpreters correctly compute the factorial of five.

```scheme
(define fact-5
  '(((lambda (!)
      (lambda (n)
        (((! !) n))))
    (lambda (!)
      (lambda (n)
        (if (zero? n)
            1
            (* n (((! !) (sub1 n))))))))
  5))

(define test-value-of
  (lambda ()
    (and
      (= (expressed-value→scheme-value (value-of/proc/proc fact-5 (base-env/proc/proc))) 120)
      (= (expressed-value→scheme-value (value-of/proc/ds fact-5 (base-env/proc/ds))) 120)
      (= (expressed-value→scheme-value (value-of/ds/proc fact-5 (base-env/ds/proc))) 120)
      (= (expressed-value→scheme-value (value-of/ds/ds fact-5 (base-env/ds/ds))) 120))))

(test-value-of)
```

In total, you should have four interpreters, five helper procedures for each, and the `expressed-value→scheme-value` helper procedure, for a total of twenty-five procedure definitions for Part I.
Part II: Logic Programming

Write the answer to the following problems using your knowledge of the Mini-Kanren logic programming system. With each answer, give a full-sentence intuitive justification as to why and how you came up with your answer. Answers with no justification receive no credit.

Remember that #s is written as succeed, #u is written as fail, ≡ is written as ==, cond\textsuperscript{e} is written as conde, run\textsuperscript{*} is written as run*, run\textsuperscript{5} is written as run 5, and any\textsuperscript{o} is written as anyo.

1. What is the value of
   \[
   \text{run}^* (q) \\
   (\equiv 5 q))
   \]

2. What is the value of
   \[
   \text{run}^* (q) \\
   (\equiv 5 q) \\
   (\equiv 6 q))
   \]

3. What is the value of
   \[
   \text{run}^* (q) \\
   \text{cond}\textsuperscript{e} \\
   ((\equiv 5 q) #s) \\
   (#s (\equiv 6 q))))
   \]

4. What is the value of
   \[
   \text{run}^1 (q) \\
   \text{cond}\textsuperscript{e} \\
   (#u #s) \\
   (#s (\equiv 5 q))))
   \]

5. What is the value of
   \[
   \text{run}^* (q) \\
   \text{cond}\textsuperscript{e} \\
   (#u #s) \\
   (#u #u))))
6. What is the value of
\[(\text{run}^2 \ q)\]
\[(\equiv 5 \ q)\]
\[(\text{cond}^e (\equiv 5 \ q))\]
\[(\equiv 5 \ q))\]

7. What is the value of
\[(\text{let} ((x \ (\text{var} \ 'x))\]
\[(y \ (\text{var} \ 'y)))\]
\[(\text{unify} \ x \ 5 \ '((,y . 4) (.x . ,y))))\]

8. What is the value of
\[(\text{let} ((x \ (\text{var} \ 'x))\]
\[(y \ (\text{var} \ 'y))\]
\[(z \ (\text{var} \ 'z)))\]
\[(\text{reify} \ (\text{walk}^* \ x \ '((,y . ,z) (.x . ,y))))\]

**Brainteaser**

The brainteaser does not count unless the rest of the assignment is attempted.

**Part I:**

Write one interpreter \textit{value-of} that accepts as additional arguments the representation-dependent helper functions (seven total arguments) for a given closure/environment representation and works the same as the interpreter above.

**Part II:**

Suppose \textit{always} and \textit{never} are defined as
\[(\text{define} \ \textit{any}^\circ \]
\[(\text{lambda} \ (g)\]
\[(\text{cond}^e\]
\[(g \ #5)\]
\[(\text{else} \ (\textit{any}^\circ \ g))))))))\]
(define always (any° #s))
(define never (any° #u))

Write the answer and give a justification to the following questions.

1. What is the value of
   \[(\text{run}^5 (q) \text{ always})\]

2. What is the value of
   \[(\text{run}^5 (q) \text{ never})\]

3. What is the value of
   \[(\text{run}^1 (q) \text{ always} \text{ #u})\]

4. What is the value of
   \[(\text{run}^1 (q) \text{ #u} \text{ always})\]

5. What is the value of
   \[(\text{run}^1 (q) (\text{cond}^e (\text{never} (\#s))))\]

6. What is the value of
   \[(\text{run}^1 (q) (\text{cond}^e (\#s) (\text{never})))\]