CS61A Lecture 24

2011-08-01
Colleen Lewis

Goals

• Increase comfort with the meta-circular evaluator (MCE)
• Identify inefficiency
• See efficiency improvement using analyze
• Connect the ideas in analyze to compiling

mce review

• You could define new functions in mce?
  – A. True
  – B. False

How many calls to mc-eval?

;;; M-Eval input:
(define (fact n)
  (if (= n 0)
    1
    (* n (fact (- n 1)))))
A) 1  B) 2  C) 3  D) 4  E) 5

Does this make any calls to mc-apply?
A) Yes  B) NO!!!
calls to mc-eval

(fact 0)
(fact 1)
(fact 1)
(fact 2)

REVIEW What is a procedure?

STk> (mc-eval '(*(lambda (x) (* x x)) '(a 3)))
(procedure (x) ((* x x)) (((a) 3)))

Global

Params: x

Body: (+ x x)

How many calls to mc-eval?

DEMO

(fact 0) 1
(fact 1) 5
(fact 2) 8

(fact 1)
PROCEDURE (x) ((* x x)) (((a) 3)))

If (= n 0) 1 (* n (fact (− n 1))))
How many calls to mc-eval?

DEMO

(fact 0) 1
(fact 1) 5
(fact 2) 8

(fact 1)

2
3

(fact 1)

3

(fact 1)

8/1/2011
(define (mc-eval exp env)
  (cond
    ((self-evaluating? exp) exp)
    ((quoted? exp) (text-of-quotation exp))
    ((variable? exp) (lookup-variable-value exp env))
    ...)
  
  (define (analyze exp)
    (cond
      ((self-evaluating? exp)
       (analyze-self-evaluating exp))
      ((quoted? exp)
       (analyze-quoted exp))
      ((variable? exp)
       (analyze-variable exp))
      ...))

Each call to mc-eval could have a lot of sub-calls!

Most didn’t depend upon the environment so I could do in advance

analyze

(define (mc-eval exp env)
  (define (analyze exp)
    (lambda (env) (analyze exp env))
  )

What is the domain and range of analyze?

A. Domain: function Range: function
B. Domain: expression Range: function
C. Domain: function Range: expression
D. Domain: expression Range: expression
E. Other

Is the domain and range correct?  A) Yes  B) No

(list representing expression) ➔ analyze ➔ STK Scheme expression

(list representing expression) ➔ analyze ➔ STK Scheme expression

(list representing expression) ➔ analyze ➔ STK Scheme expression

(list representing expression) ➔ analyze ➔ STK Scheme expression
(define (mc-eval exp env)
  (cond ...
    ((variable? exp) (lookup-variable-value exp env)) ...
  (define (analyze exp)
    (cond ... ((variable? exp) (analyze-variable exp)) ...
  (define (analyze-variable exp)
    ;; write this!
    (lambda (env) (lookup-variable-value exp env)))

Is the thing you returned entirely scheme (it only needs to be interpreted by STk)?
  A) Yes  B) No  C) ???

(analyze-quoted)

(define (mc-eval exp env)
  (cond ...
    ((quoted? exp) (text-of-quotiation exp)) ...
  (define (text-of-quotiation exp) (cadr exp))

  (define (analyze exp)
    (cond ... ((quoted? exp) (analyze-quoted exp)) ...

  (define (analyze-quoted exp)
    (lambda (env) (text-of-quotiation exp)))

  (define (analyze exp)
    (cond ... ((if? exp) (analyze-if exp env)) ...

  (define (analyze-if exp env)
    (if (true? (mc-eval (if-predicate exp) env))
      (mc-eval (if-consequent exp) env)
      (mc-eval (if-alternative exp) env)))}
(analyze '(if #t 3 4))
(define (analyze exp)
  (cond ((self-evaluating? exp)
          (analyze-self-evaluating exp))
       ((if? exp) (analyze-if exp))...)
(define (analyze-if exp)
  (let ((pproc (analyze (if-predicate exp)))
     (cproc (analyze (if-consequent exp)))
     (aproc (analyze (if-alternative exp))))
  (lambda (env)
    (if (true? (pproc env))
        (cproc env)
        (aproc env))))

Do we save time using the analyzing mce?
(sent-sum '(1 2 3 4 5 6 7 8))
A) Yes  B) No  C)??

Do we save time using the analyzing mce?
(sent-sum '())
A) Yes  B) No  C)??

Do we save time using the analyzing mce?
(list (+ 2 3) (+ 4 5) (+ 2 3))
A) Yes  B) No  C)??

Do we save time using the analyzing mce?
(list (sq 2) (sq 3) (sq 4))
A) Yes  B) No  C)??

Compiling Java

Compiling Java

Compilers

• Analyze syntax
• Make something that can be run on a computer
• Provide optimization
• Provide useful feedback to the programmer when there are errors

Environments (below the line)

Environments (below the line)
How many calls to \texttt{mc-eval}?

\\texttt{;;; M-Eval input:}
\begin{verbatim}
(define (simple x) x)
\end{verbatim}

A) 1  B) 2  C) 3  D) 4  E) 5

\begin{verbatim}
(define (mc-eval exp env)
  (display (list 'mc-eval 'exp: exp))
  (newline)
  (cond ((self-evaluating? exp) exp)
        (mc-eval exp: (define (simple x) x))
        (mc-eval exp: (lambda (x) x)))
\end{verbatim}

Is the thing you returned entirely scheme (it only needs to be interpreted by STk)?  A) Yes  B) No  C) ???