SICP cons-stream

Asked 6 years, 1 month ago Active 6 years, 1 month ago Viewed 2k times

2

1

regarding to SICP 3.5

my own implementation is as following

```
(define (delay exp) (lambda () exp))

(define (force delayed-obj)
  (delayed-obj))

(define (cons-stream a b) (cons a (delay b)))

(define (stream-car stream) (car stream))


(define (stream-cdr stream) (force (cdr stream)))

(define (take n stream)
  (if (= n 0)
      (print "0")
      (begin (take (- n 1) (stream-cdr stream))
             (print n))))

(define (make-stream-enum-interval low high)
  (if (> low high)
      '()
      (begin (print low) (cons-stream low (make-stream-enum-interval (+ low 1)
high)))))
```

actually I found it's not really delayed. when I execute (define range-10-to-100 (make-stream-enum-interval 10 100)). I expect to see only 10 is print in the console. while it's 10.....100

is there anything wrong with my code? Or, if print 10...100 is necessary, then we can say the structure is (cons 10 (delay cons 11 (delay cons 12 (delay ....100))) if so, then we need more memory?

scheme sicp cons

asked Jul 2 '14 at 10:52

user3754786
47 1 5

You can also use lips macros `(define-macro (delay . exp) `(lambda () ,@exp))` – jcubic May 22 '19 at 7:23

1 Answer

Active Oldest Votes

Scheme uses eager evaluation. That means that all arguments to a function call are evaluated

8

before the function is entered. Thus, since your `delay` was a function, expressions passed to `delay` get evaluated first.

To get around this, make `delay` a macro, as does anything that uses it (like `cons-stream`). Here's a reformulation of your `delay` and `cons-stream` as macros:

```
(define-syntax delay
  (syntax-rules ()
    ((_ exp) (lambda () exp))))

(define-syntax cons-stream
  (syntax-rules ()
    ((_ a b) (cons a (delay b)))))
```

answered Jul 2 '14 at 11:09

Chris Jester-Young
199k 43 361 408