CS 61A Week 12

Topic: Metacircular evaluator

Midterm Wednesday , 7–9pm.

Reading:

Abelson & Sussman, 4.1.1–6

MapReduce paper in course reader.
(metacircular evaluator: `~cs61a/lib/mceval.scm`)

Homework:

Some students have complained that this week's homework is very time-consuming. Accordingly, with some reluctance, I've marked a few exercises as optional; these are the ones to leave out if you're really pressed for time. But it's much better if you do all of them! The optional ones have * next to them.

1. A&S exercises 4.3, 4.6, 4.7*, 4.10*, 4.11*, 4.13, 4.14, 4.15

2*. Modify the metacircular evaluator to allow *type-checking* of arguments to procedures. Here is how the feature should work. When a new procedure is defined, a formal parameter can be either a symbol as usual or else a list of two elements. In this case, the second element is a symbol, the name of the formal parameter. The first element is an expression whose value is a predicate function that the argument must satisfy. That function should return `#t` if the argument is valid. For example, here is a procedure `foo` that has type-checked parameters `num` and `list`:

```
> (define (foo (integer? num) ((lambda (x) (not (null? x))) list))
    (list-ref list num))
FOO
> (foo 3 '(a b c d e))
D
> (foo 3.5 '(a b c d e))
Error: wrong argument type -- 3.5
> (foo 2 '())
Error: wrong argument type -- ()
```

In this example we define a procedure `foo` with two formal parameters, named `num` and `list`. When `foo` is invoked, the evaluator will check to see that the first actual argument is an integer and that the second actual argument is not empty. The expression whose value is the desired predicate function should be evaluated with respect to `foo`'s defining environment. (Hint: Think about `extend-environment`.)

Extra for experts:

Abelson & Sussman, exercises 4.16 through 4.21

Unix feature of the week: `echo`, `set`, `setenv`, `printenv`

Emacs feature of the week: `M-!` (run shell command)