

6.001 Tutorial 8

David Ziegler

November 1, 2004

1 Administrivia

- Problem set 7 is due tomorrow.
- Project 3 is due Friday.

2 Environment Model Rules

The environment model is an accurate way of thinking about how computation in Scheme works — it's not a lie! There are five simple rules:

Name rule The value of a name n in an environment e is the binding for n in the first frame of e where n is bound.

define rule A define expression creates or replaces a binding for the name in the first frame of e .

set! rule A set! expression changes the binding of n in the first frame of e where n is bound.

lambda rule Create a double bubble whose environment pointer is e .

Apply rule This is the important one! To apply a procedure p ...

- Create a new frame f .
- Set f 's parent pointer to be the same as p 's environment pointer — they are “hand-cuffed” together.
- In f , bind the parameters of p to the values of the arguments it was applied to.
- Evaluate the body of p in the newly created environment.

3 Memoization

A useful optimization that some languages perform is *memoization*. The idea of memoization is to figure out when a function returns a constant answer for a given set of arguments. If it does, whenever we call a function, we remember the arguments and the result. Later, we can check if we've seen these arguments before and return the result without doing the computation again. For expensive functions (lots of work), this can save a lot of time. The following procedure takes a single argument function and returns a memoized version.

```
(define (memoize proc)
  (let ((vals '()))
    (lambda (arg)
      (let ((previous (assoc arg vals)))
        (if previous
            (cadr previous)
            (let ((temp (proc arg)))
              (set! vals
                    (cons (list arg temp)
                          vals))
              temp))))))
```

Remember that `assoc` is a procedure that operates on association lists. It compares the `car` of each element of the list to the first argument and returns that element if it is. It has the following behavior:

```
(assoc 5 '((5 25) (6 36)))
; Value: (5 25)
```

```
(assoc 3 '((5 25) (6 36)))
; Value: #f
```

```
(assoc 3 '())
; Value: #f
```

Now imagine we memoize a simple procedure:

```
(define fast-square  
  (memoize (lambda (x) (* x x))))
```

```
(fast-square 5)
```

What happens in the environment?