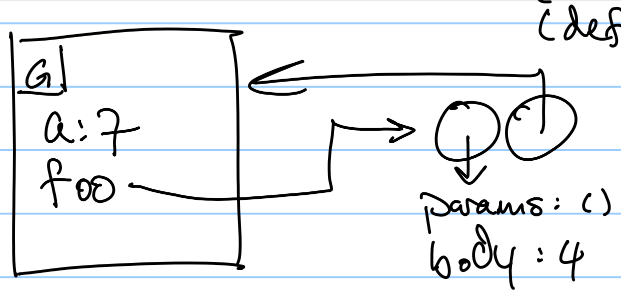
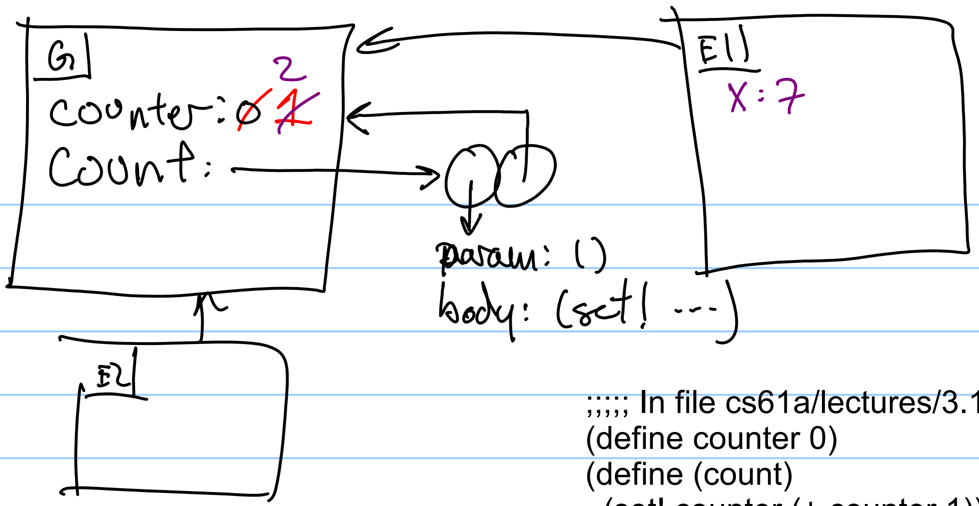
Note Title

7/14/2010

```
(define a 7)
(define foo (λ () 4))
```



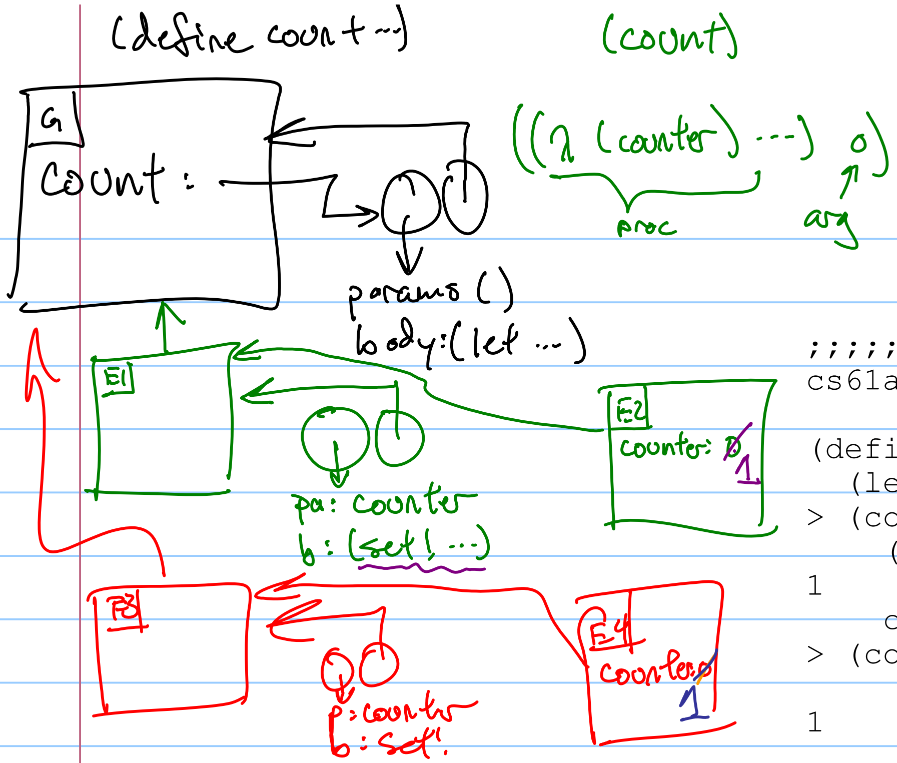


```
;;;;; In file cs61a/lectures/3.1/count1.scm
(define counter 0)
(define (count)
  (set! counter (+ counter 1))
  counter)
> (count)
1
> (count)
2
```

```
(let ((var1 val1)
      (var2 val2))
   body)
```

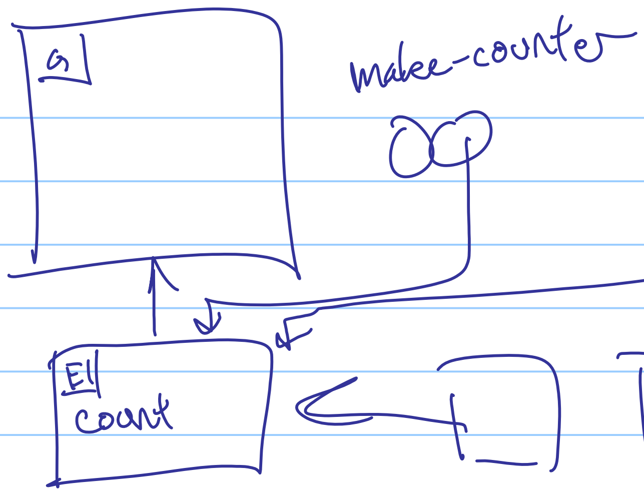
$$\Updownarrow$$

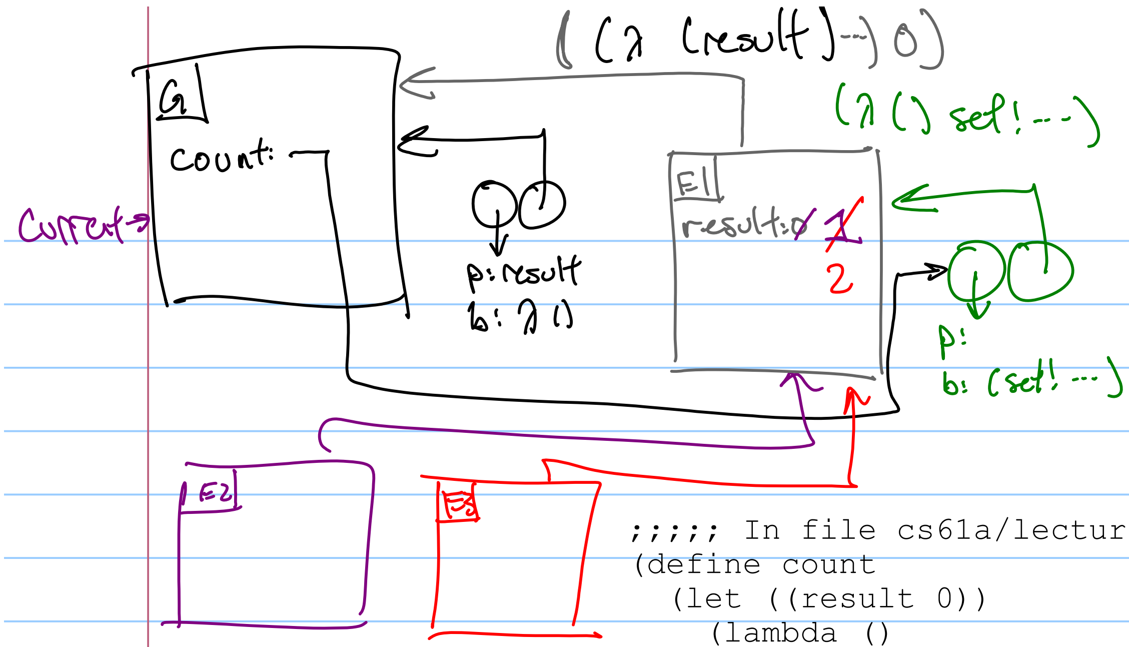
```
((λ (var1 var2) body) val1 val2)
```



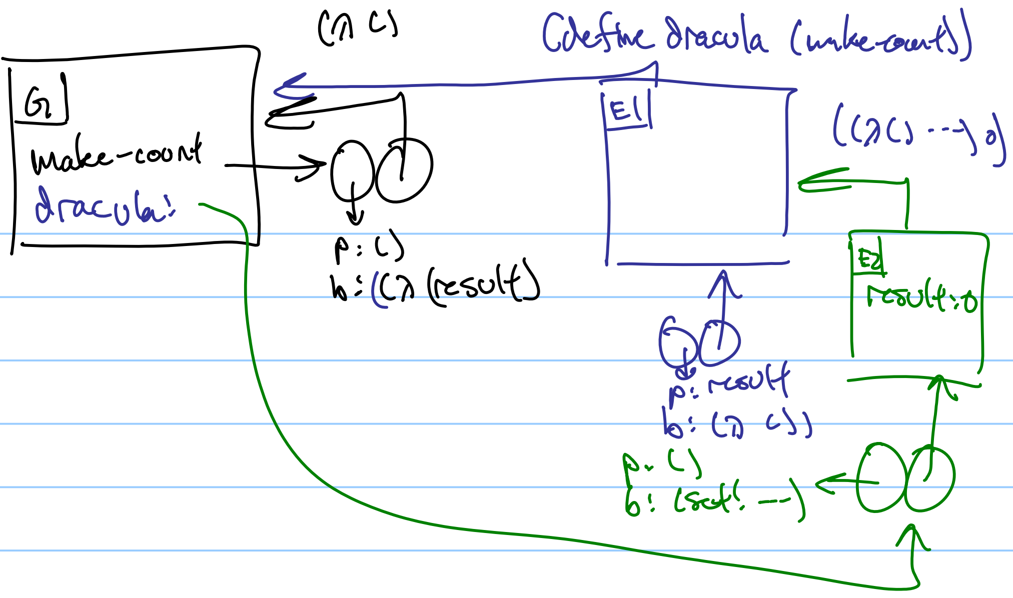
```
;;;;; In file
cs61a/lectures/3.1/count.lose

(define (count)
   (let ((counter 0))
> (count)
      (set! counter (+ counter 1))
1
      counter))
> (count)

1

> (count)
```

What we want





```
;;;;; In file cs61a/lectures/3.1/count2.scm
(define count
  (let ((result 0))
    (lambda ()
      (set! result (+ result 1))
      result)))
```



```
;;;;; In file cs61a/lectures/3.1/count3.scm
(define (make-count)                          > (define dracula (make-count))
  (let ((result 0))                           > (dracula)
    (lambda ()                                1
      (set! result (+ result 1))              > (dracula)
      result)))                               2
```