The Environmental Model of Execution

Amitabha Sanyal

Department of Computer Science and Engineering
IIT Bombay
Mumbai - 400076

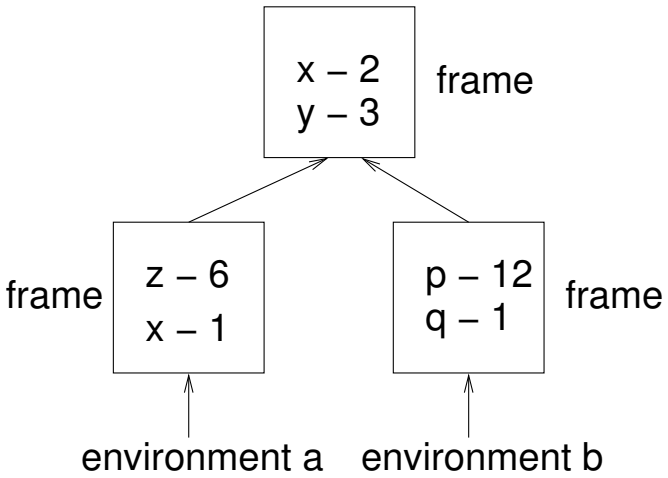
as@cse.iitb.ac.in

March 2012

Environment model of execution

Main idea – Each function must carry its own environment

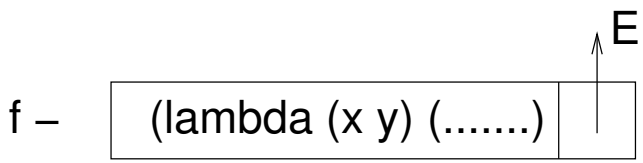
What does the environment look like?



- A define extends a frame.
- A function call or a let creates a new frame.
- An environment is a chain of frames.

Environment model of execution

- A function name is bound to a pair of things:
  - A lambda
  - An environment



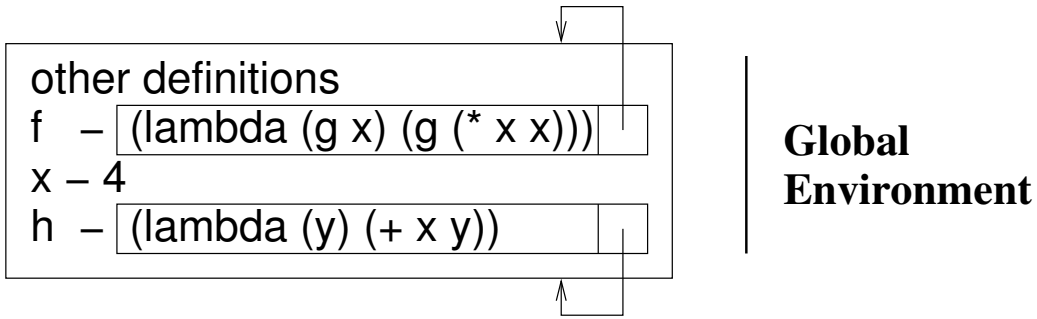
The environment is the one in which the function definition was being processed.

Building the environment

Rule 1: `defines` extend the current frame

Rule 2: The environment of a lambda is the current environment at the point where the lambda is being evaluated.

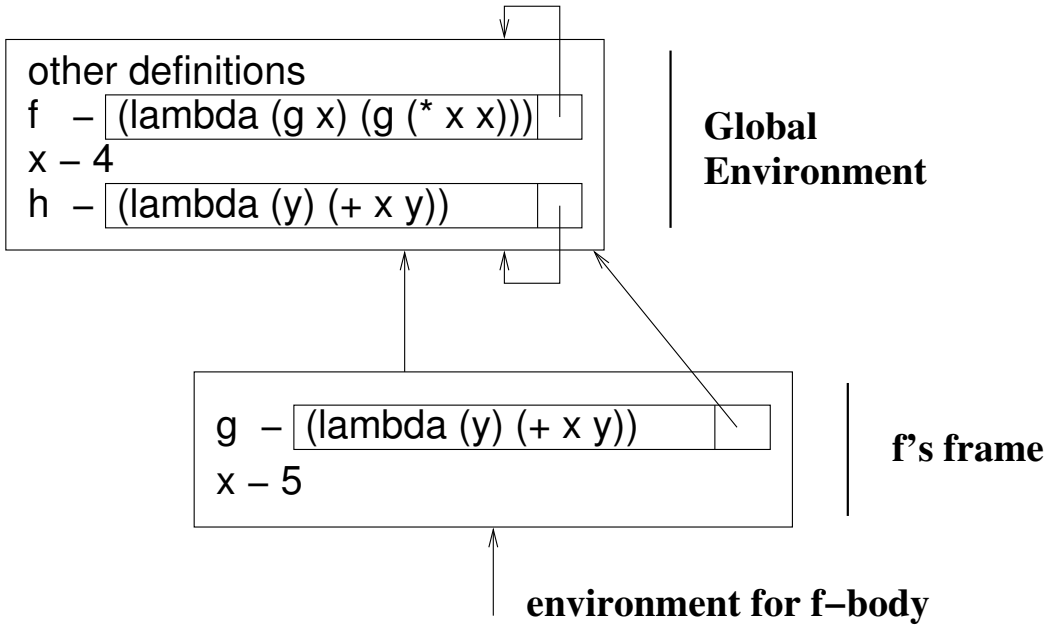
```
(define (f g x) (g (* x x)))
(define x 4)
(define (h y) (+ x y))
(define w (f h 5))
```



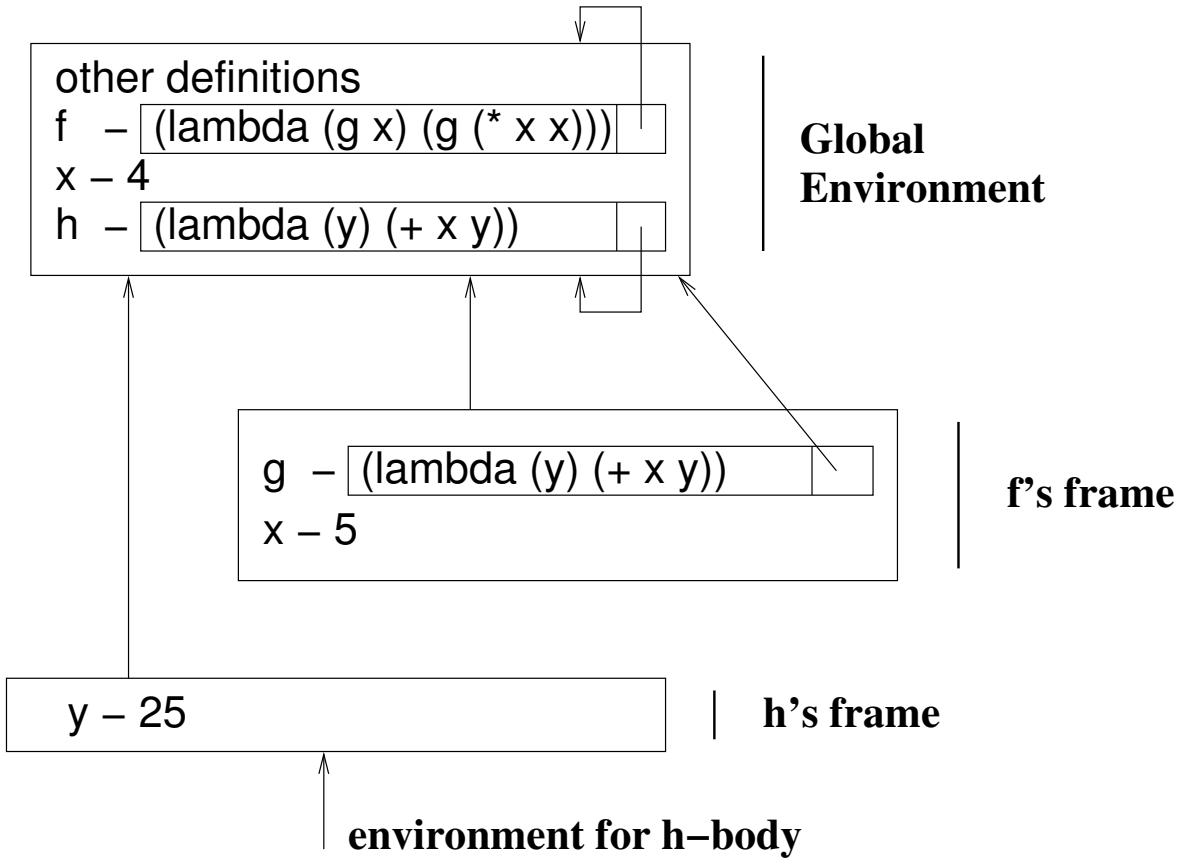
Building the environment

Rule 3: During a function call, a new frame containing the parameters is created.

Rule 4: The global pointer of the function frame is made to point to the environment carried by the lambda of the function.



First example

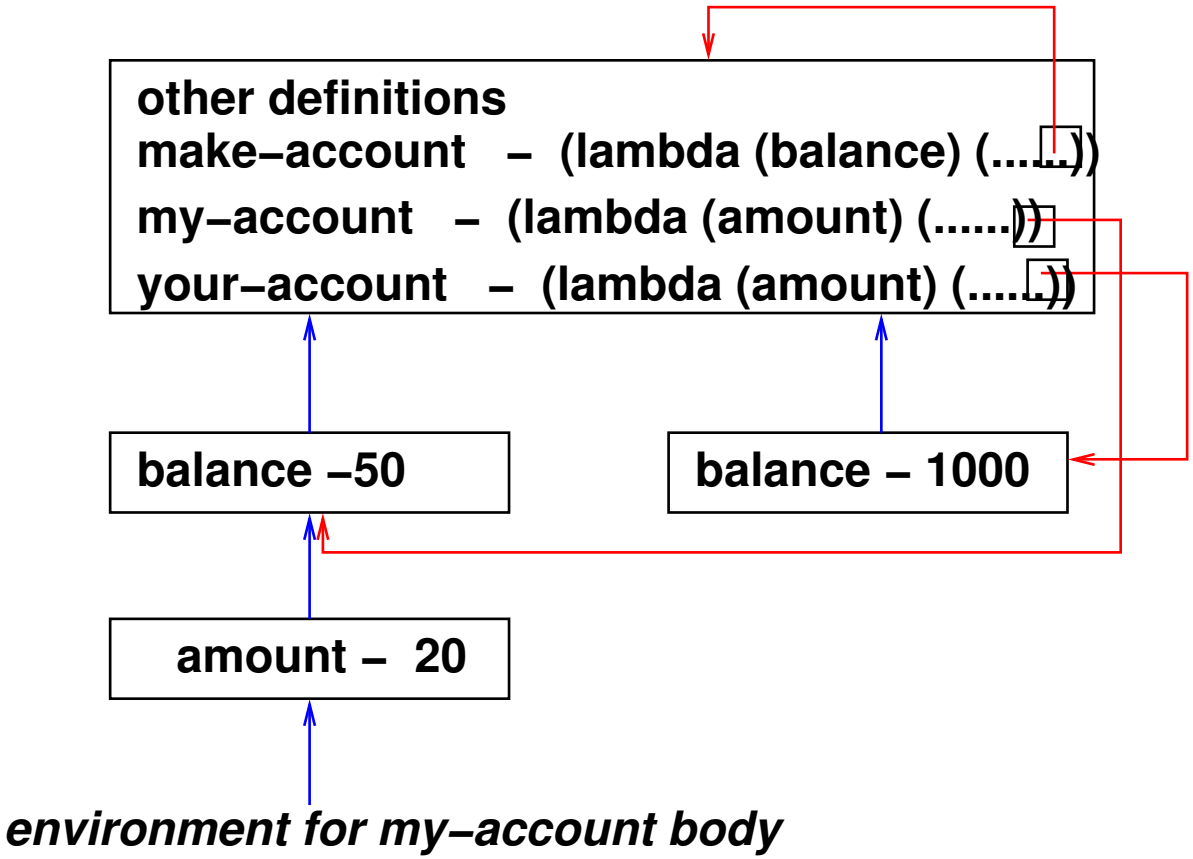


A second example

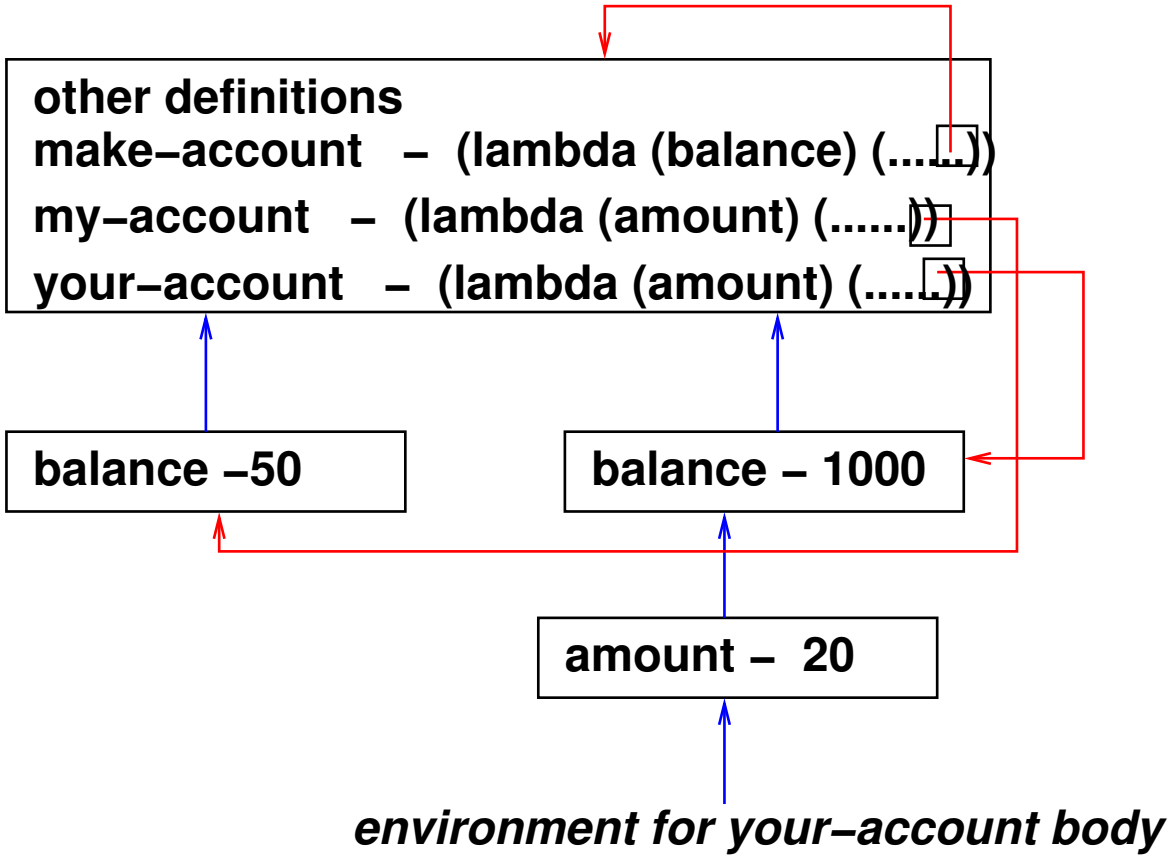
```
(define (make-account balance)
   (lambda  (amount)
      (if (>= balance amount)
          (begin
             (set! balance (- balance amount))
             balance)
          "Insufficient funds")))

(define my-account (make-account 50))
(define your-account (make-account 1000))
1 ]=> (my-account 20)
;Value: 30
1 ]=> (your-account 20)
;Value: 980
1 ]=> (my-account 50)
;Value 2: "Insufficient funds"
```

Second example

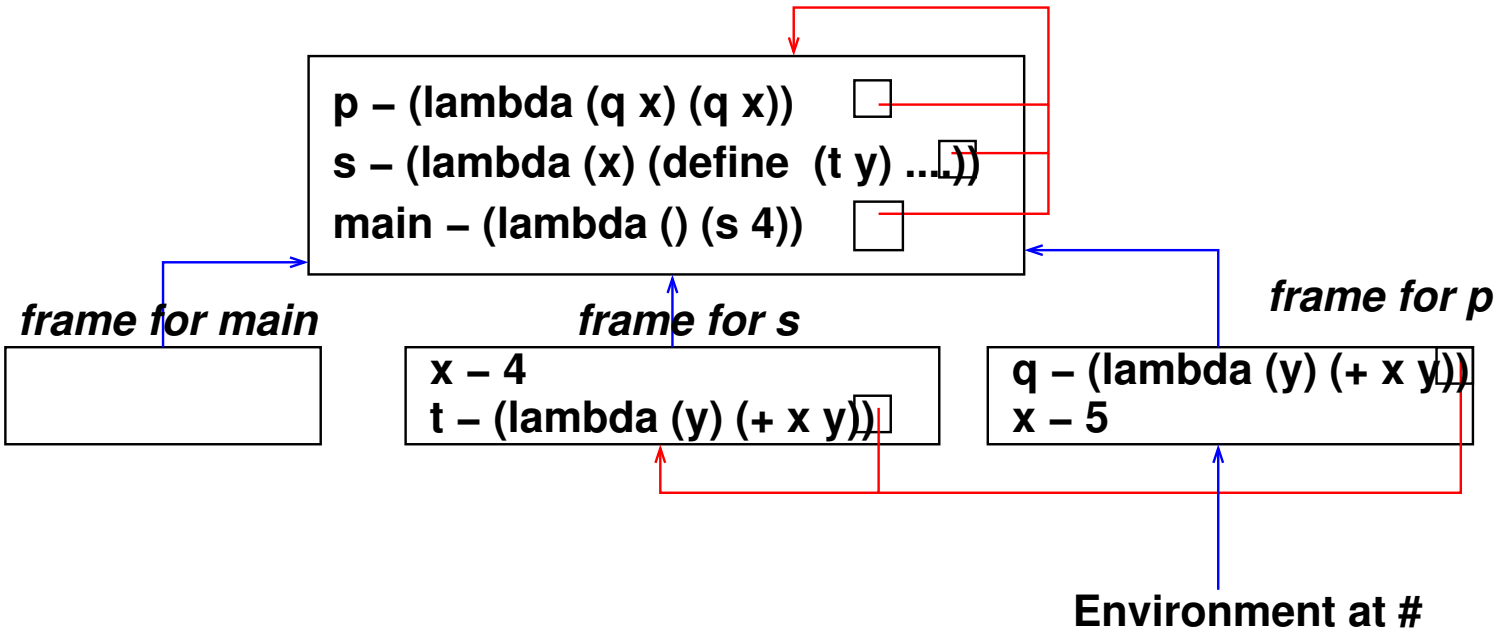


Second example



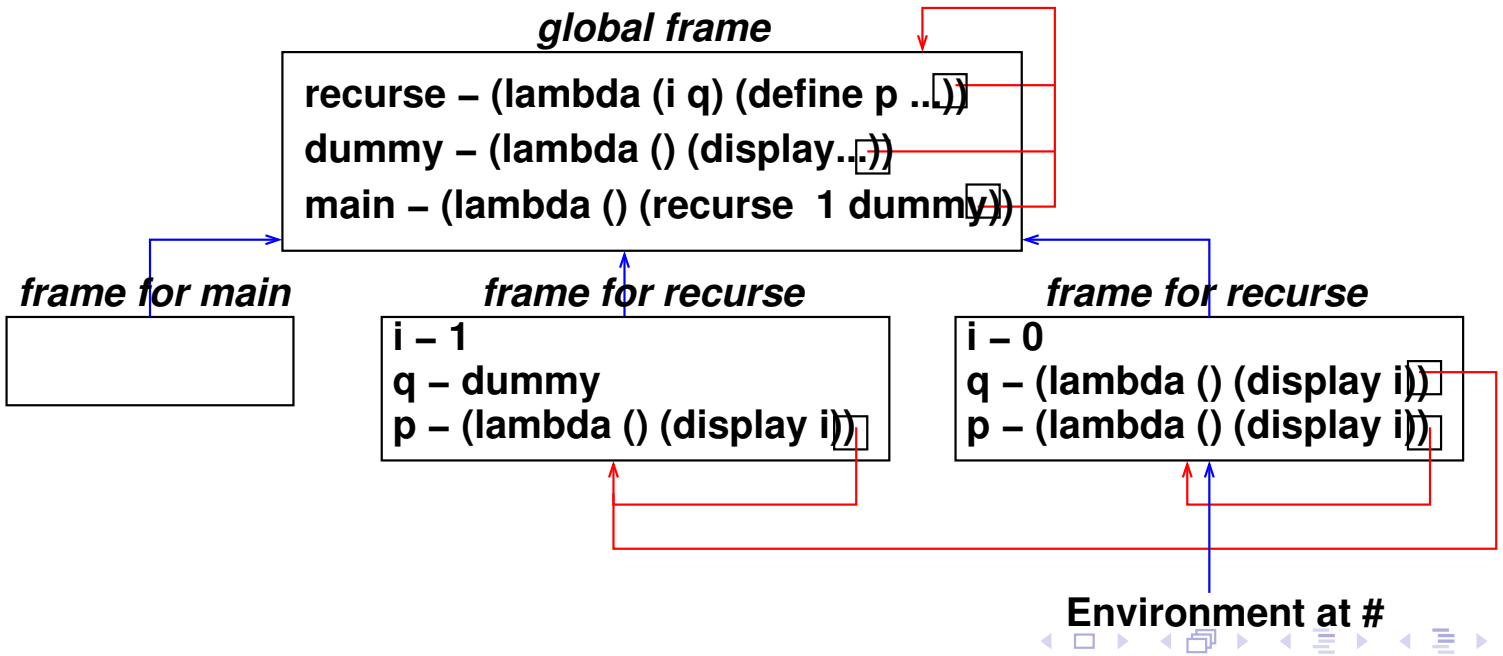
Third example

```
(define (p q x) #(q x))
(define (s x)
  (define (t y) (+ x y))
  (p t 5))
(define (main) (s 4))
(main)
```



Fourth example

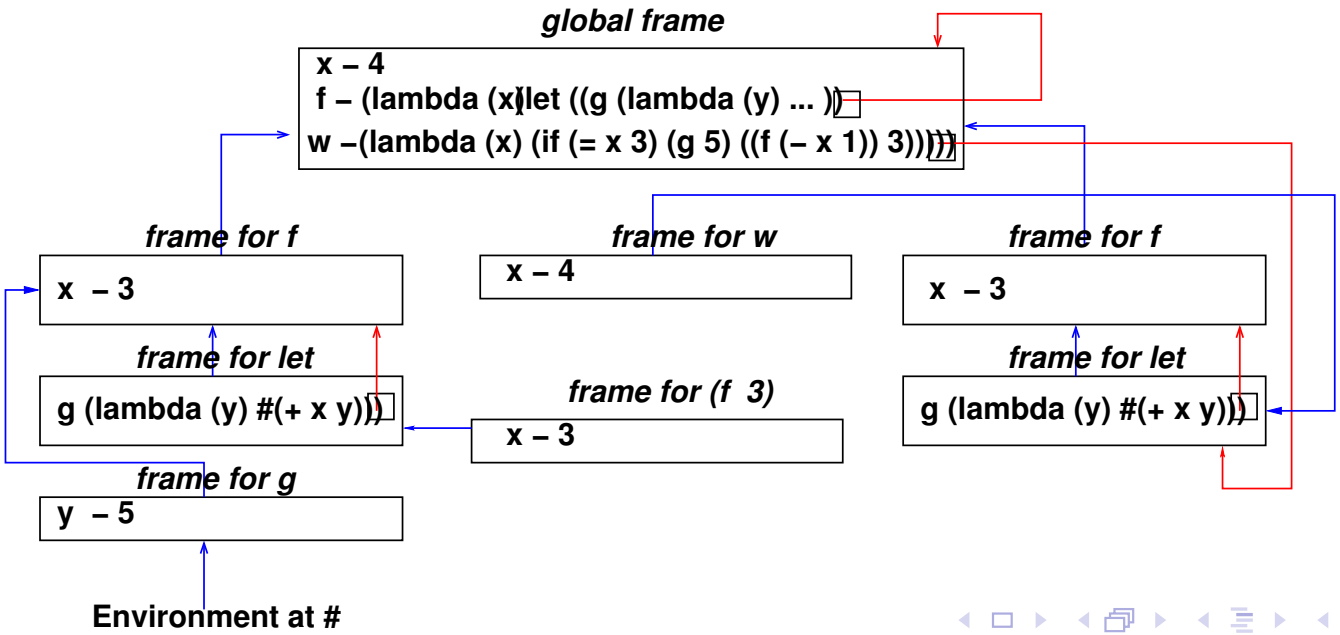
```
(define (recurse i q)
  (define (p) (display i))
  (if (> i 0) (recurse (- i 1) p)
      (begin # (p) (q))))
(define (dummy) (display ""))
(define (main) (recurse 1 dummy))
(main)
```



Fifth example

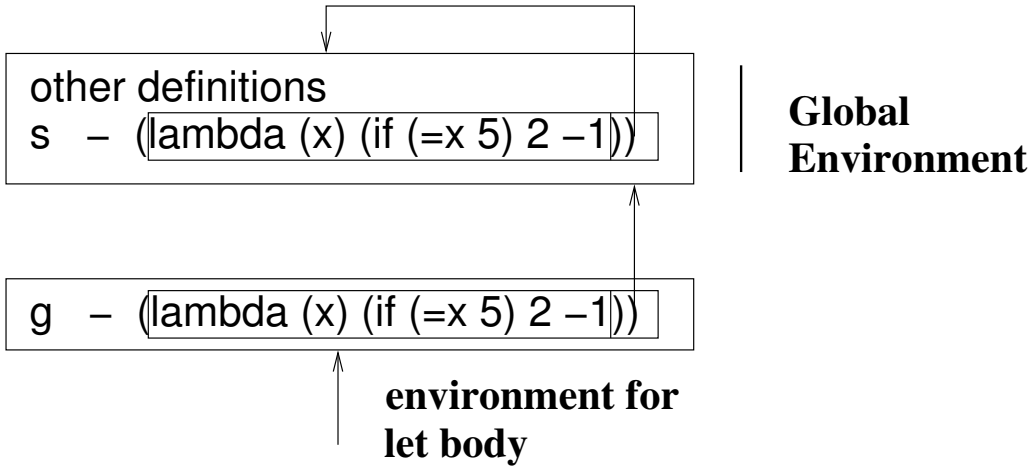
Rule 5: A `let` creates its own frame.

```
(define x 4)
(define (f x)
  (let ((g (lambda (y) #(+ x y))))
    (lambda (x) (if (= x 3) (g 5) ((f (- x 1)) 3)))))
(define w (f 3))
(define result (w 4))
```



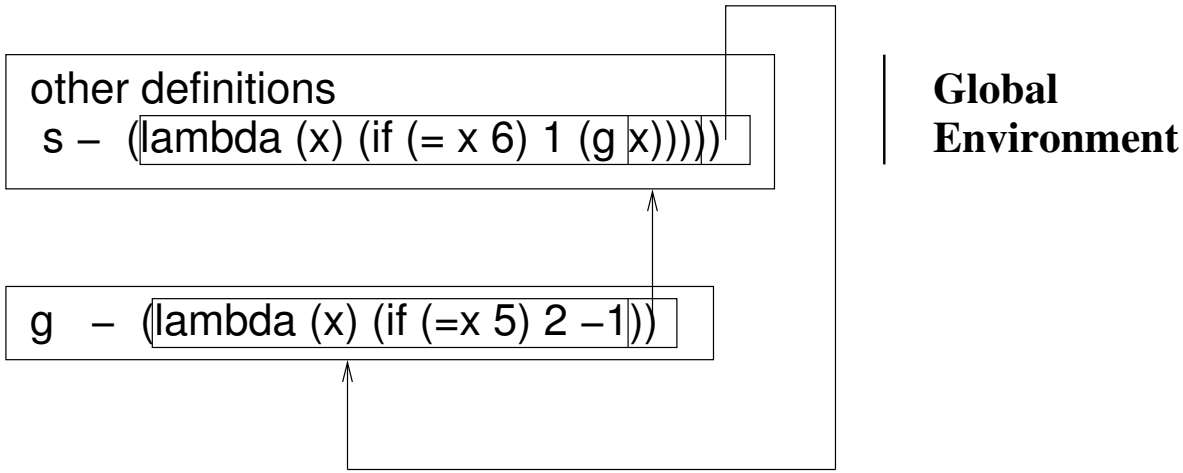
Sixth example

```
(define (s x) (if (= x 5) 2 -1))
(set! s (let ((g s)) (lambda (x) (if (= x 6) 1 (g x)))))
```



Sixth example

After the `set!`



Seventh example

```
(define (s x) (if (= x 5) 2 -1))
(set! s (lambda (x) (if (= x 6) 1 (s x))))
```

After the `set!`