

(A (value of expr env)  $\star$  T  
ST cov.)

(number expr)

[ n #: when (number? n) n ]

[ R1 #: when (symbol? R1) (box (apply env env)) ]

(box (apply env env))

[ (x) (b) ]

(X (a) (value of (extend-env x a env)))

[ (set! y e) (set-box (apply env env))  
(value of e env) ]

[ (factor #) #: when (symbol? #) (R1 (number? #) (box (apply env env))) ]

(value of (factor env) (box (apply env env)))

[ (factor, rand) ]

((value of (factor env) (box (value of rand env))))

$(\lambda \text{ (value of expr env) } \underline{\text{CBO}})$

$(\text{quote } \text{envs})$

$\llbracket n \text{ \# : when (numbers? n) } n \rrbracket$

$\llbracket R' \text{ \# : when (symbols?) } \text{ (box } \times \text{open)} \rrbracket$

$\llbracket (\text{box } \times \text{open}) \text{ (apply-env env y)} \rrbracket$

$\llbracket f(x)(b) \rrbracket$

$\llbracket (\lambda (a) (\text{value of } (\text{extend-env } x \text{ a env}))) \rrbracket$

$\llbracket (\text{set. } y, e \text{ (set-box)} (\text{apply-env env y}) (\text{value of } e \text{ env})) \rrbracket$

$\llbracket (\text{factor } \#) \text{ \# : when (symbol?) } \rrbracket$

$\llbracket (\text{value of } \text{factor env}) \rrbracket$

$\llbracket (\text{factor } \text{rand}) \rrbracket$

$\llbracket (\text{value of } \text{factor env}) (\text{box } (\text{value of } \text{rand env})) \rrbracket$

$\llbracket (\text{apply-env env y}) \rrbracket$

"CB Reference" Yes

Can modifying parameters in a subroutine affect the calling procedure?

"CBV

No  
;; create a local

parameter to subr.

```
((lambda (x) ; create a local parameter to subr.
  (begin (set! y 7)
          x))
  )
```

} Subroutine

5)

X → 5 } CBV behavior

CBU: 5 y → 7

CBR: 7 x → 7 } CBR behavior  
y → 7

CBV

alias in parameters to  
subroutines?

CBR

strict

Not

?

Strict

do evaluate

cbname | cbv

the args

to  
BH closure

do we evaluate the argument before  
applying the function?

(A value expr env) (CBName)

(nater env)

[ n #: when (number?) n ]

[ #: when (symbol?)  
R ]

[ (R na env) (apply) ]

[ (x) (y) ]

[ (x as (value) extend-env x a env) ]

[ (R #: when (symbol?)  
R) ]

(env) (value) (R)

[ (R na env) ]

[ (factor) ]

[ (value) (factor) (env) (base) (R) (value) (R) ]

$\rightarrow ((\lambda(x) 120) \leftarrow$

$\Omega$ )

IN OPEN 120  $\nabla$ .

$\Omega = ((\lambda(x) (x x)) (\lambda(x) (x x)))$

inf loop.

$\lambda(\lambda(x) 120)$

$((1 \text{ lots}))$

$((\lambda(x) (+ x x x x x x))$

$((1 \text{ bang}))$

(A (valof expr env) (C3Hane C3BneeR

(muten eyes

[ n #: when (numbers? n) ]

[ #: when (symbols?) ]

(unbox-need apply-env env y)

[ (x) (y) ]

(X as (valof b (extend-env x a env)))

(R: (symbols) when #: (R: (symbols))

(valof (factor) (env))

[ (factor, (rand)) ]

((valof (factor) (env)) (box(x) (valof (rand) (env))))

$(\Delta (\text{unbox-need } b)$   
 $(\text{let } ((\text{unbox } b))))))$

(begin

(set-box! b ( $\lambda()$  val)) ←

val))

memoization



[Random, next] (Random (valof next end))

> ((x(x) (+ x x)))

(Random 121)

in cb need, value of this is even