Interpretación:

- variables
- funciones

```
(evaluate '((lambda (a) a) 1)
          ())
```

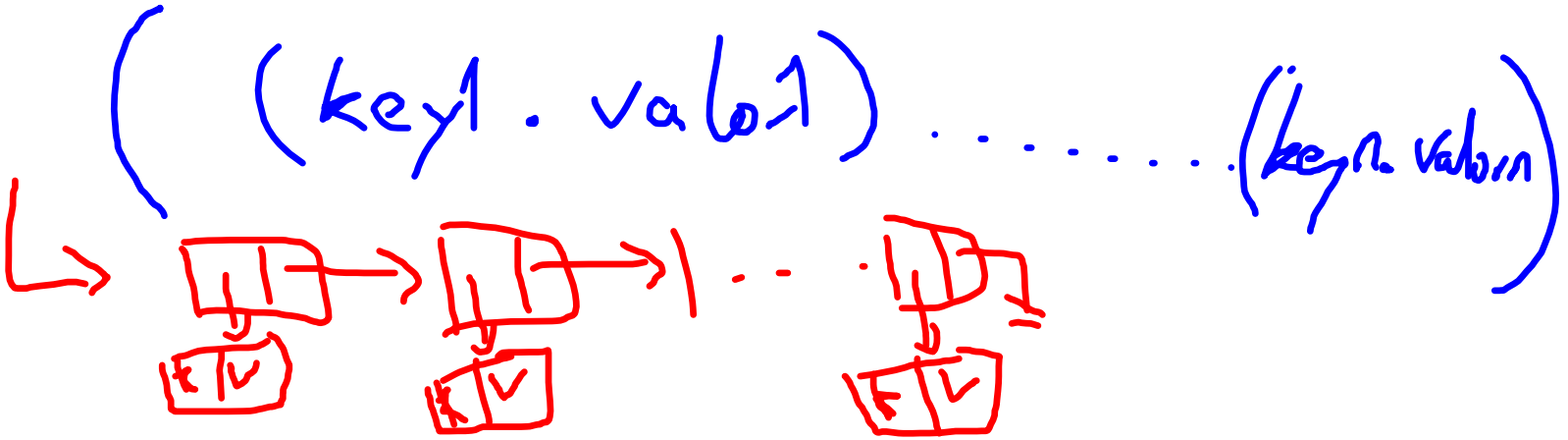
→ 1

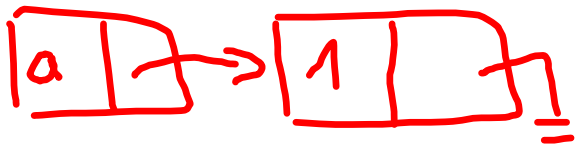
env:

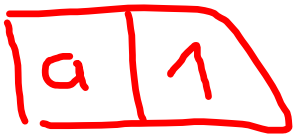
key ↔ value

asociación variable ↔ valor

↳ lista de asociación:



'(a 1) → 

'(a . 1) → 

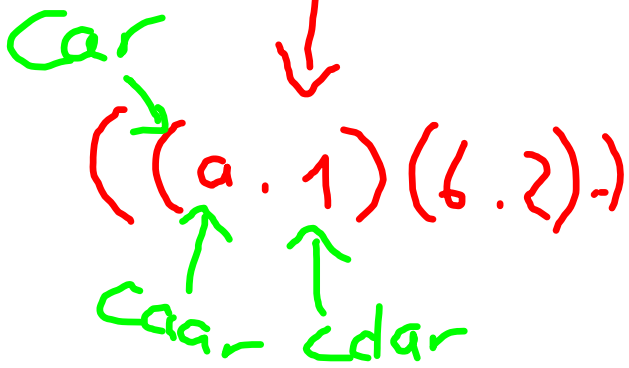
(evaluate 'a '((a . 1)))

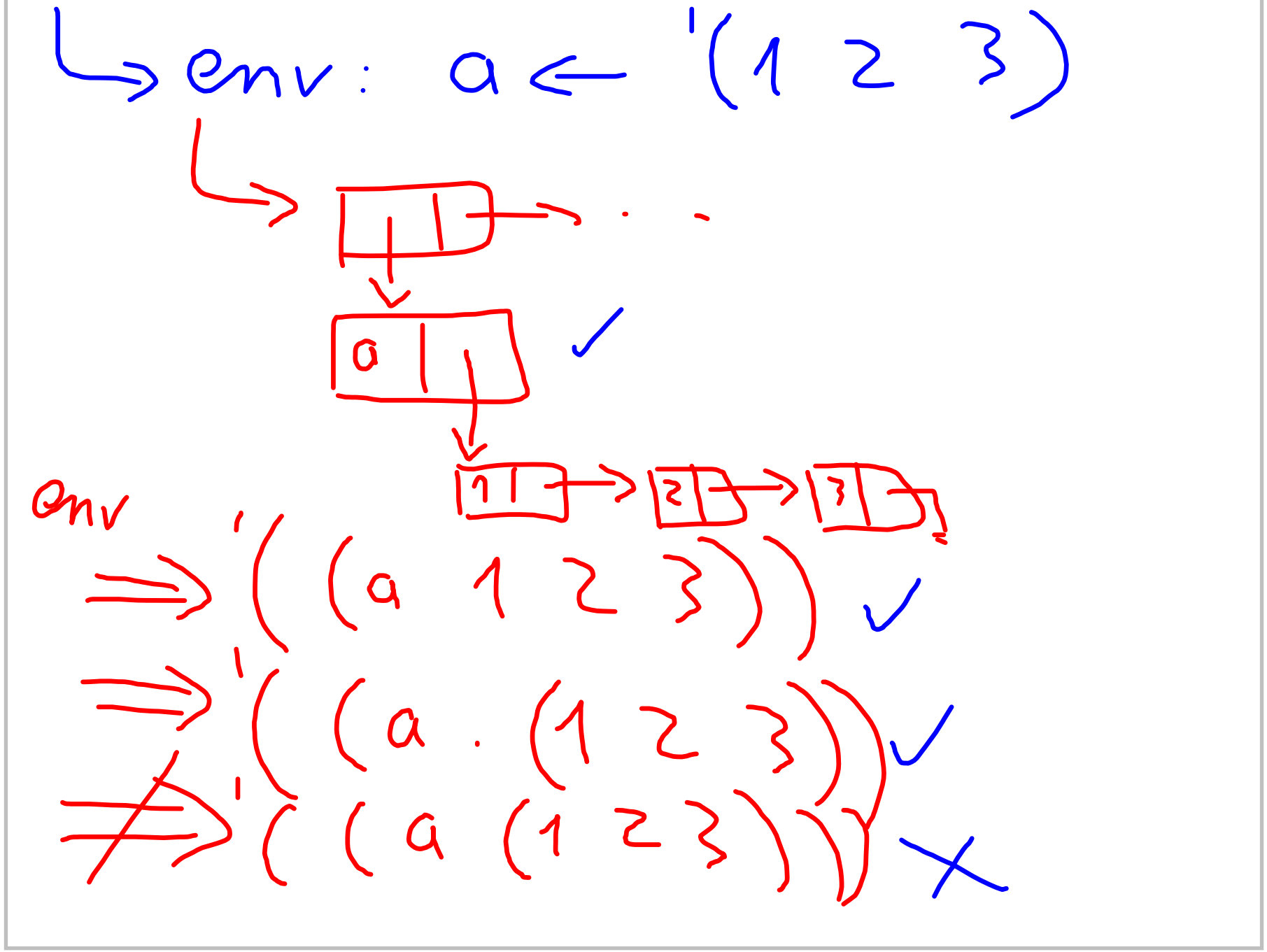
↳ 1

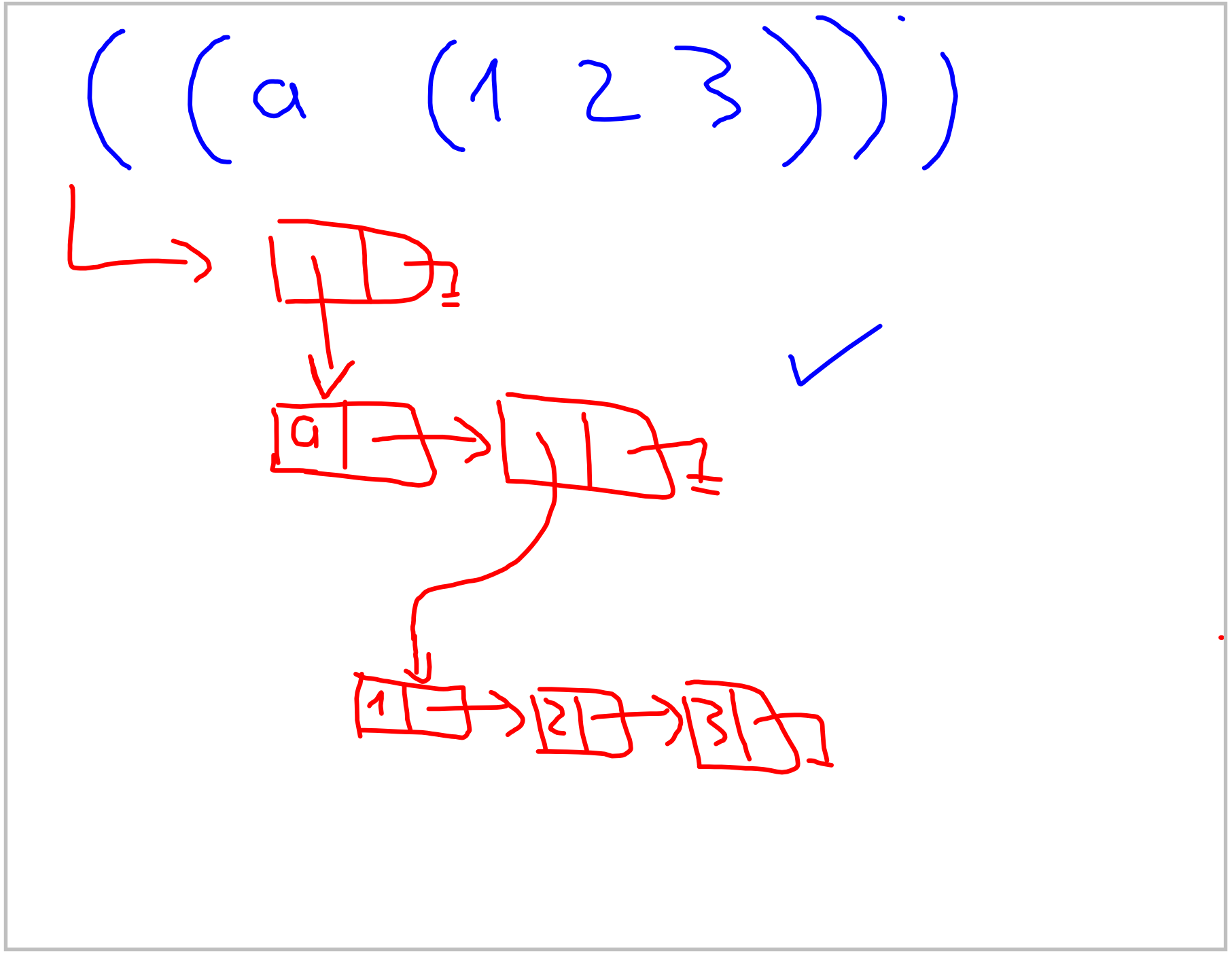
(evaluate 'a '())

↳ undef

```
(define (lookup var env)
  (if (null? env)
      'undef
      (if (eq? var (caar env))
          (cdar env)
          (lookup var (cdr env)))))
```




env: a ← '(1 2 3)
a
1
2
3
env
'((a 1 2 3))
'(a . (1 2 3))
'(a (1 2 3))


((a (1 2 3)))
a
1
2
3

↳ invoke:

¿qué es un valor-funcional?

invoke sepa llamar una función creado con make-function

invoke: {funcion
{lista de values

```
(define (invoke f args)
  (if (procedure? f)
      (f args)
      (wrong "not a function")))
```

(lambda (a b) (+ a b))

variables → '(a b)

body → '((+ a b))

↳ evalie body

↳ en a ← valo
b ← valo + env