tuples

Monday, 28 February 2022 8:33 AM

(define v (vector 1 2 3))

(vector-ref v 0) → 1

(vector-set! v 0 5) → (5 2 3)

mutation

(define v (1 #t 3))

int → bool

type :: Int | Bool | (Vector type+) |
void

exp :: = if | with |
(vector exp+) |
(vector-ref exp int) |
(vector-set! exp int exp) |
(void)

index value

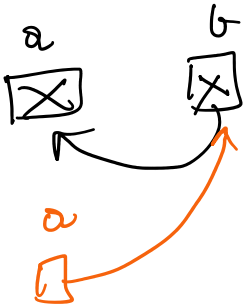
properties

1) first class.

(let ([t (vector 40 #t (vector 2))]) t)

2) aliases

a = b →

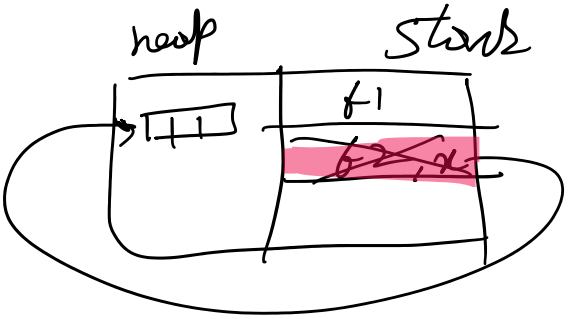


(let ([t1 (vector 3 7)])
(let ([t2 t1])

(let ([_ (vector-set! t2 0 42)])
  (vector-ref t1 0)))

↙

42



3) life of tuples → alive

compilers
user

a) safe →

b) not so safe → garbage collector

live | dead

↙ ↘

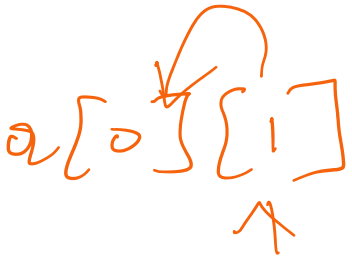
access | lost access

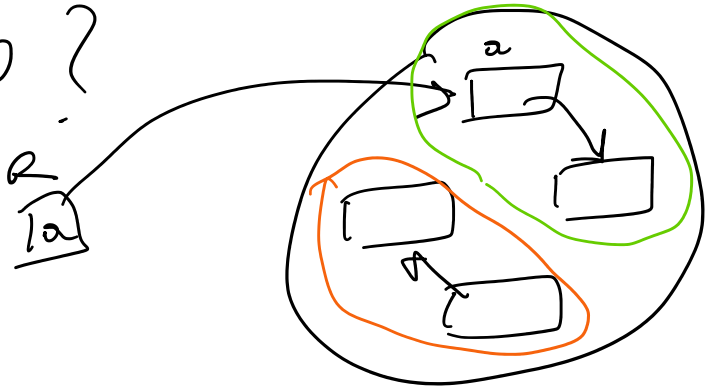
pointer to tuples →

→ register.

→ stack.

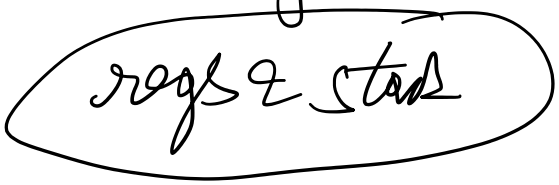
→ heap ?

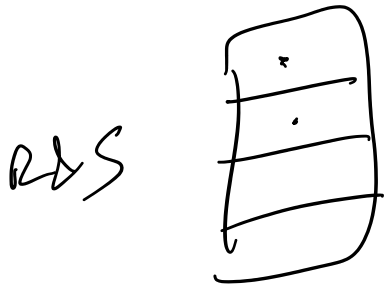


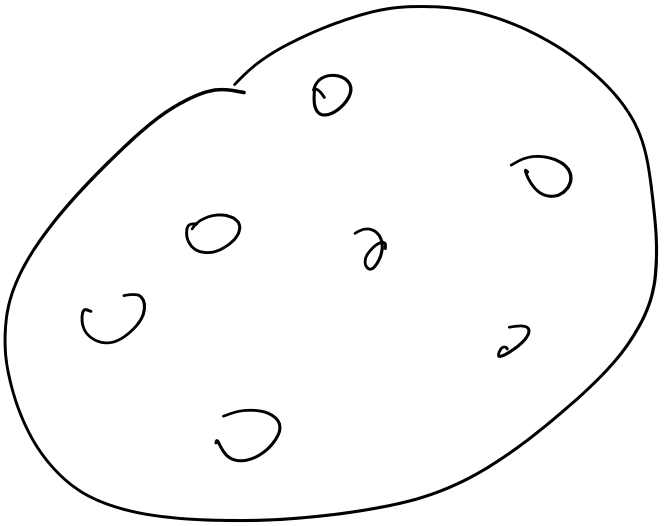


regs or stack

transitively check for liveness







Garbage collector

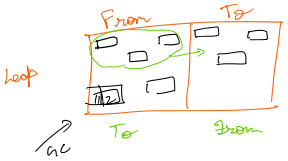
Monday, 28 February 2022 9:13 AM

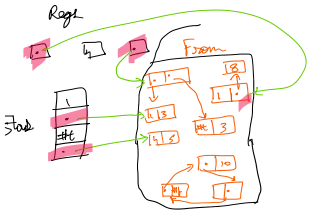
2 space garbage collector → copy collector

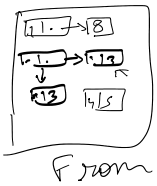
mark & sweep

reference counter

generational







Step 1

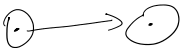
a roots pointers → copied to To space

Step 2

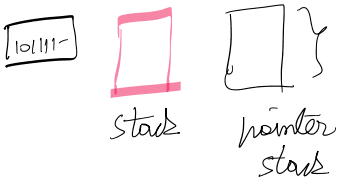
iterative over each root pointer/ copied data

Step 3

update the address



Identify pointers / vector names ?

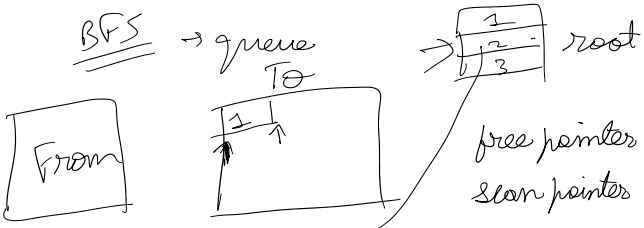


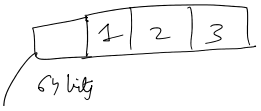
-8(%rbp)

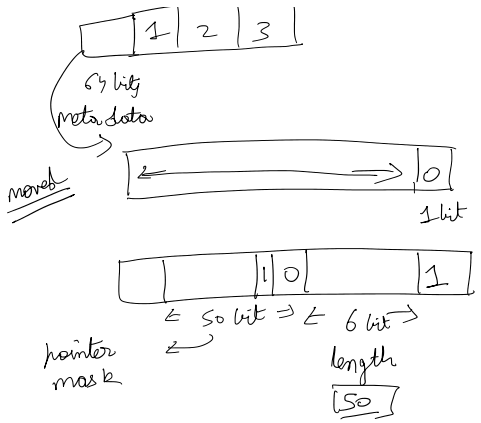
16 Int & Bool

4 vector

BFS → queue

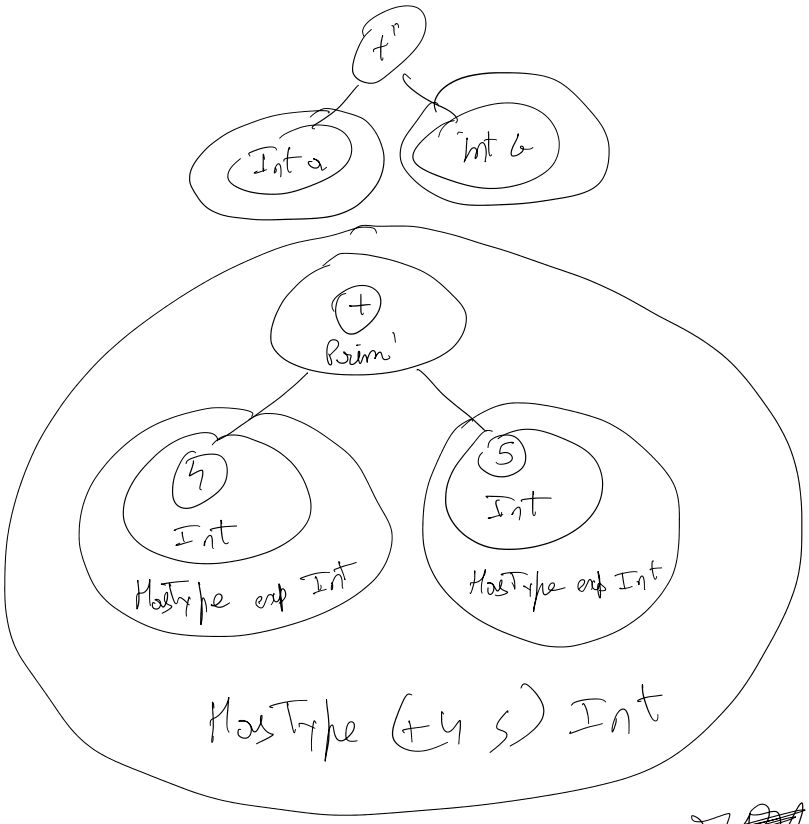






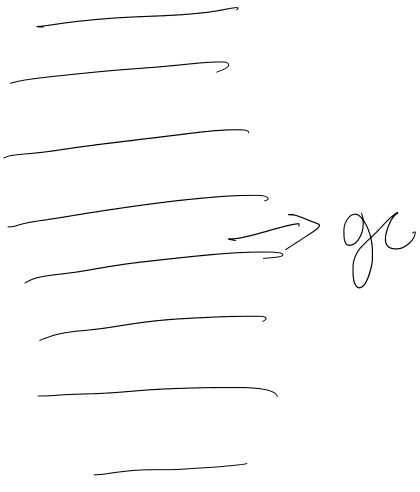
Regs & stack

(prim +' ( ) ( ))



vector

gc



v ~~room~~

v ↔ room