10-1 (let ([x 7])
((let ([y 8]) ...) (+ (let ([y 8]) y))-R

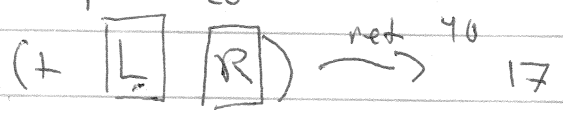
uniquify

y is illegal))

flatten 10)

!y = 8
!let = !y
!r = !let + 10

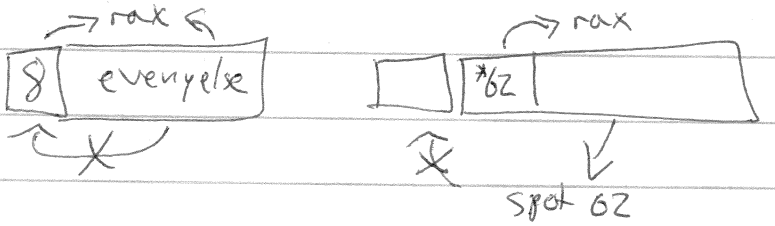
liveness



x = alloca (17) rsp ← rsp - 17

free [x] rsp ← rsp + 17

sucks less, ory



Ty += (Vector Ty ...) // (Vector int int)
(Void) (Vector (Vector int int)
(V int int))

Expr += (vector e ...)
(vector-ref e int)
(vector-set! e int e)
(void)

$\Gamma \vdash e : t \ldots$

$\Gamma \vdash$ (vector e ...) : (Vector t ...)

$\Gamma \vdash e$: (Vector $t_0 \ldots t_n$)

$\Gamma \vdash$ (vector-ref e i) : t_i

$\Gamma \vdash e$: (Vector $t_0 \ldots t_n$) $\Gamma \vdash a : t_i$

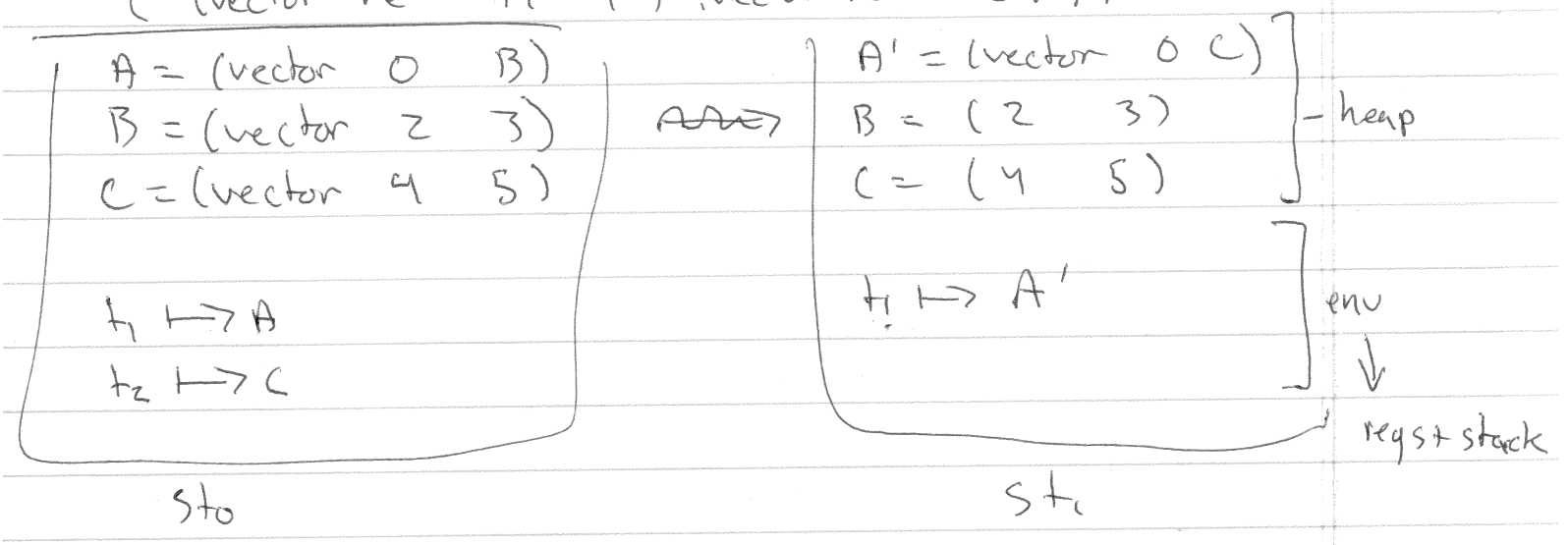
$\Gamma \vdash$ (vector-set! e i a) : (void)

(let ([t_1 (vector 3 7)])
(let ([t_2 t_1])
(let ([_ (vector-set! t_2 0 42)])
(vector-ref t_1 0))))

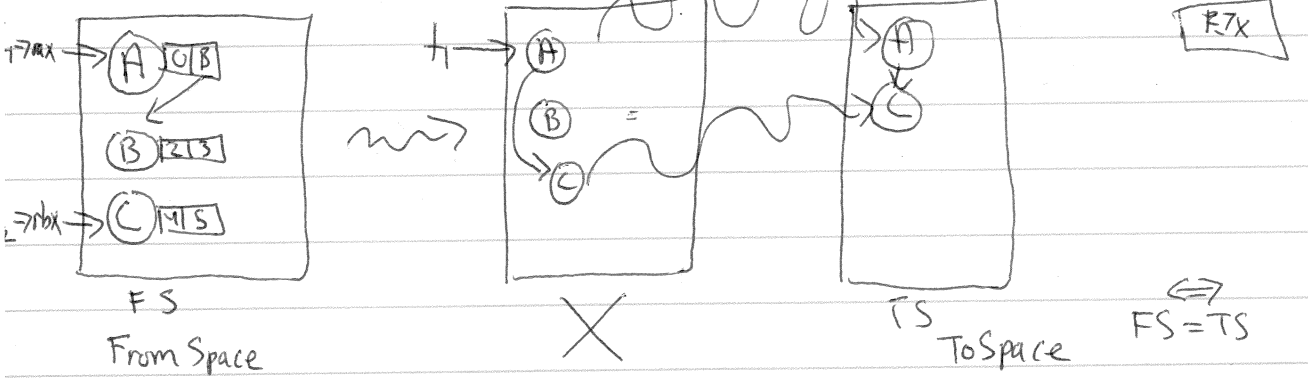
} begin

=> ?

10-2 (let ([t_1 A (vector 0 B)]) (let ([t_2 C (vector 4 5)]) (begin (vector-set! t_1 1 t_2)) ((vector-ref t_1 1) vector-ref 0)))



Stop & Copy



In (regs ∪ stack), which are pointers?
Mapping from Ptr to Ty ⇒ compiler only stores ptrs on stack

What about cycles? NEXT TIME