10-1

Tuples (Vectors) (Structs)

↳ A collection of sub-components
↓
not all the same thing

↳ int x int x bool
struct { int x; int y; bool z; }

↳ first-class
(int x int) x int

↳ tuples have identity and are mutable
⇒ not FIFO extent ⇒ heap-allocated

↳ tuples are not pointers (there's no free) ⇒ GC
(not numbers)

T = ... | (Vector T ...) | Void
E = ... | (vector E ...)
| (vector-ref E int)
| (vector-set! E int E)
| (void)

(and E_1 E_2) := (if E_1 E_2 #f)
(or E_1 E_2) := (if E_1 #t E_2)
(begin E_1 E_2 ... E_n) := (let ([_ E_1])
(begin E_2 ... E_n))
(begin E_1) := E_1
(begin) := (void)
(when E_c E_1 ... E_n) := (if E_c (begin E_1 ... E_n) (void))
(unless E_c E_1 ... E_n) := (when (not E_c) E_1 ... E_n)

10-2/

```
(vr (vr (vec 1 (vec 23) 4)
        1)
    0)
=> 2        (first-class)
```

```
(let ([x 5]) (let ([y (+ 1 x)])
  (vec x y 7)))
=> (vec 5 6 7)   : (Vector Int Int Int)
```

```
(let ([t1 (vec 3 7)])
(let ([t2 t1])                          ' vector-eq? '
 (begin (vs! t2 0 42)
        (vr t1 0))))    => 42
```

$$\frac{\Gamma \vdash E_i : T_i}{\Gamma \vdash (\text{vec } E_0 \ldots E_n) : (\text{Vec } T_0 \ldots T_n)}$$

$$\frac{\Gamma \vdash E : (\text{Vec } T_0 \ldots T_N)}{\Gamma \vdash (\text{vr } E\ k) : T_k} \qquad \frac{\Gamma \vdash E_1 : (\text{Vec } T_0 \ldots T_N) \quad \Gamma \vdash E_2 : T_k}{\Gamma \vdash (\text{vs! } E_1\ k\ E_2) : \text{Void}}$$

$$\Gamma \vdash (\text{void}) : \text{Void}$$

Type Checker : Expr => Type

⇓

Typer : Expr → Expr^T

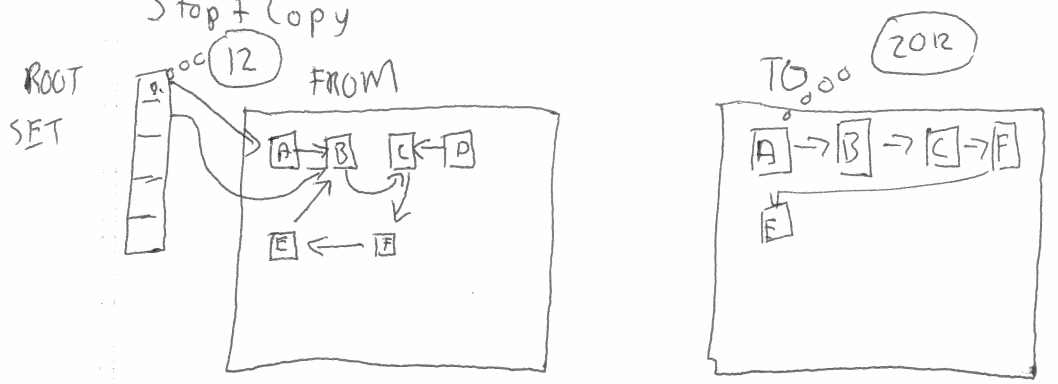
E = V | N | (+ E E) | (if E E E) | B

E^T = (TypeIs T E^-)

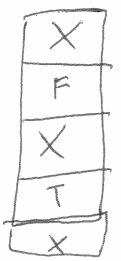
E^- = V | N | (+ E^T E^T) | (if E^T E^T E^T) | B

```
(let ([v1 (vector 0 ... 3000000))])
  :
  (vector-ref  v1  24)
  :
  (vector-set! v1  32)
  :
  (+ 4 4))
```

Stop + Copy



|F| = |T|



when running (ASM) — r/ws to FROM
when GC runs — r/ws to FR & TO
swap

mprotect
mmap

GC: traverse root set
explore obj refs
remember what was copied
update root set / obj refs

```
(vr (let ([x (vec 1 2)])
      x)
    0)
```

Cheney Copying

Queue of objs to look at = To Space
Head = To Sp Tail = To Sp
Enq = Put it on the Tail (step 1. Enq the root set)
Loop: when head → tail,
  copy (enq) the obj refs
  update head