19-1 Automatic Sound Memory Management — Garbage Collection

~~"When variables leave scope, they may be freed."~~

What can be freed?

Reachability

reach(o) iff var(o)

$\vee$ ptr(p, o) $\wedge$ reach(p)

$\vee$ field(o', o) $\wedge$ reach(o')

$\vee$ reg(o)

$\vee$ stack(o)

unreachable objs may be freed!

IM-2/ CEK = ⟨expr, env : var ⇀ val, kont⟩

CESK = ⟨ctrl, env : var ⇀ loc, sto : loc ⇀ val, loc⟩

⟨ v, env, kapp (clo(λx. e, env'), env'', k) ⟩
↦ ⟨ e, env'[x ↦ v], k ⟩

⟨ σ_1, env, sto, σ_2 ⟩ ↦ ⟨ σ_6, env'[x ↦ σ_7], σ_4 ⟩

where sto(σ_1) = v

sto(σ_2) = kapp (σ_3, env'', σ_4)

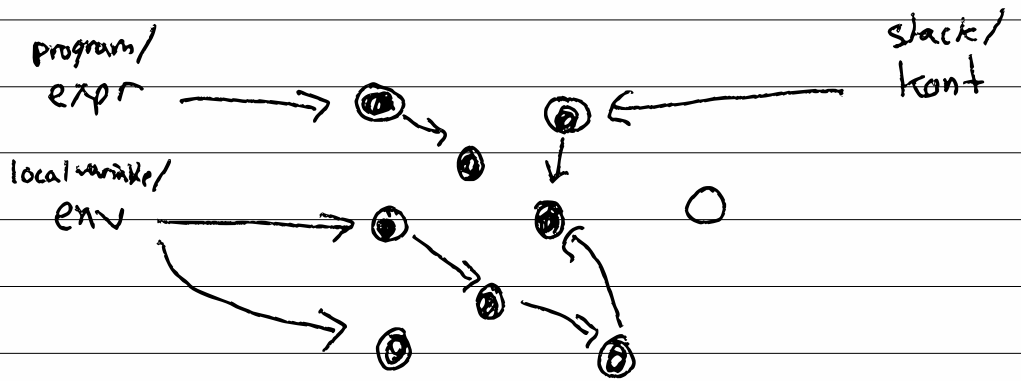
sto(σ_3) = clo(σ_5, env')

sto(σ_5) = lam (x, σ_6)

sto' = sto [σ_7 ↦ v]

19-31

reachable (<σe, env, sto, σk>)



Mark + Sweep (globals, locals, stack)

- first trace all objects from the root set, marking
- walk all memory, delete things w/o a mark, or unmark

19-4/ given a pointer, what is it pointing to?

↳ GC needs to know the structure of mem

how do you inspect the stack?

how do you mark?

↳ Bi Bo P

Mark + Sweep

~~NOT~~ optimal for time

optimal for space

optimal for latency

malloc — O(lg n)

free — ~~O(lg n)~~

mark — O(live)

sweep — O(live + dead = mem)

space overhead — free list + marks + tags

pauses/latency — real-time incremental strategy exists