18-1

Mutation — variables that vary like C/etc

M ::= ... | (set! X M)

((λx. (begin (set! x 5) x)) 6) ⇒ 5

((λx. (set! x 5)) 6) ⇒ 5 (begin M N) ::= ((λ_. N) M) N

(something not used)

" int x = 6;
x = 5
return x; "

(begin 1 2) = ((λ_. 2) 1)
⇒ 2

((λx. (begin (set! x 5) x)) 6)
⇒ (begin (set! 6 5) 6)
⇒ ((λ_. 6) (set! 6 5))
⇒ ((λ_. 5) 5)
⇒ 5

mutation is incompatible w/ substitution

" int x = 6;
x = 5;
ret x "

" 6 = 5 ⇒ ret 6 "

((λx.
(+ M
N))
42)

N's value is independent of M's execution (w/o mutation)

M := ... | σ | (set! σ M)

P := (mem ([σ V] ...) M)

ISWIM + global variables (no pointers)

E := ... | (set! σ E)

(mem ([σ_x V_x] ...) E[(λx. M) V])
→ (mem ([σ_x V_x]) E[M[x ← V]])

(mem ([σ_0 V_0] ... [σ_j V_j] [σ_{j+1} V_{j+1}] ...) E[σ_j])
→ (mem ([σ_0 V_0] ... [σ_j V_j] [σ_{j+1} V_{j+1}] ...) E[V_j])

(mem ([σ_0 V_0] ... [σ_j V_j] [σ_{j+1} V_{j+1}] ...) E[(set! σ_j V_{new})])
→
(mem ([σ_0 V_0] ... [σ_j V_{new}] [σ_{j+1} V_{j+1}] ...) E[V_{new}])

V := ... | σ

M := ... | deref M | set! M N

E := ... | deref E | set! E N | set! V E

global vars + pointers

(mem Σ E[(λx. M) V]) → (mem Σ E[M[x ← V]])

(mem Σ E[deref σ_x]) → (mem Σ E[Σ(σ_x)])

(mem Σ E[set! σ_x V_n]) → (mem Σ′ E[V_n])

$\Sigma' = \Sigma[\sigma_x \to V_n]$

V := ... | $\hat{\lambda}\sigma x$. M

(mem Σ E[($\lambda\sigma x$. M) V]) → (mem Σ′ E[M[x ← σ_n]])

$\Sigma' = \Sigma[\sigma_n \to V]$ and σ_n is new

18-3/

"int x = 5;
x = 6;
x = x + x;
ret x;"

→ "int x = 6;
x = x + x;
ret x"

→ "int x = 6;
x = 6 + x;
ret x"

→ "int x = 6;
x = 6 + 6;
ret x"

→ "int x = 6
x = 12
ret x"

→ "int x = 12
ret x"

→ "int x = 12
ret 12"

→ 12

(vars ... E[f(v)]) →
(vars ... E[(block (fs local vars) [x=v] fns body)])

E[(block (vars ... (return v))]
→ E[v]

for (init; cond; step) { body }
:= init; while (cond) { body; step }

E[while C B] → E[if C (B; while C B) void]

"C"

m := ... | σ | (set! σ m) S = Store
E := * | E[X → V]
K := ret | fn(N, E, K) | ar(V, K) | set(σ, K)

CESK

⟨ σ, E, K ⟩ → ⟨ V, E, K ⟩ V = ???

S := * | S[σ ↦ V]
⟨ σ, E, S, K ⟩ → ⟨ V, E, S, K ⟩ V = S(σ)
⟨ V, E, S, set(σ, k) ⟩ → ⟨ V, E, S', K ⟩ S' = S[σ → V]
⟨ set! σ m, E, S, K ⟩ → ⟨ m, E, S, set(σ, k) ⟩
⟨ m N, E, S, K ⟩ → ⟨ M, E, S, fn(N, E, k) ⟩
⟨ V, E, S, fn(N, E', k) ⟩ → ⟨ N, E', S, ar(V, k) ⟩
⟨ V, E, S, ar(clo(λx.m, E'), k) ⟩ → ⟨ m, E'[x→v], S, k ⟩

18-4 $((\lambda x.((\lambda y.(\text{deref } x))\ (\text{set! } x\ 5)))\ 6) = \underset{\textcircled{2}}{\textcircled{1}}\ 6$

$\langle \textcircled{1}, *, *, ret \rangle$

$\langle (\lambda x.((\lambda y.\text{deref } x)\ (\text{set! } x\ 5)))\ , *, *, fn(6, *, ret) \rangle$

$\langle clo(\textcircled{2}, *)\ , *, *, \text{"} \rangle$

$\langle 6\ , *, *, ar(clo(\textcircled{2}, *), ret) \rangle$

$\langle ((\lambda y.\text{deref } x)\ (\text{set! } x\ 5))\ , E_1 = *[x \to \sigma_1], *[\sigma_1 \to 6], ret \rangle$

$\langle (\lambda y.\text{deref } x)\ , E_1\ , \text{"}\ , fn((\text{set! } x\ 5), E_1, ret) \rangle$

$\langle clo(\uparrow, E_1)\ , E_1\ , *[\sigma_1 \to 6]\ , fn(\text{set! } x\ 5, E_1, ret) \rangle$

$\langle \text{set! } x\ 5, E_1\ , *[\sigma_1 \to 6]\ , ar(clo(\lambda y.\text{deref } x, E_1), ret) \rangle$

$\langle 5\ , E_1\ , *[\sigma_1 \to 6]\ , set(\sigma_1, ar(\quad)) \rangle$

$\langle 5\ , E_1\ , *[\sigma_1 \to 5]\ , ar(\quad) \rangle$

$\langle \text{deref } x, *[x \mapsto \sigma_1][y \mapsto \sigma_2], *[\sigma_1 \to 5][\sigma_2 \to 5], ret \rangle$

$\langle \sigma_1\ , \text{"}\ , \text{"}\ , ret \rangle$

$\langle 5\ , \text{"}\ , \text{"}\ , ret \rangle$

$\to 5!$

---

w/o mutation

$mean(+\ M\ N) = f(mean(M), mean(N))$

w/ mutation

$mean(+\ M\ N) = f(mean(M), mean(effect(M)))$

C: `int x = 7;` `int f(int* x) { return 0; }`

`return (f(&x) + x);` `int f'(int* x) { *x = 99; return 1; }`

mutation = sequential computation

no mutation = parallel

$\langle M\ N, E, K \rangle$ proc... $\to \langle M, E, send(ch_1) \rangle\ \langle N, E, send(ch_2) \rangle$

$\langle app(wait\ ch_1)(wait\ ch_2), E, K \rangle$ proc...