9-1/ ISWIM: normal reduction (included symmetry)
standard reduction (had 1 choice for work)
CC-machine (directed the search for work)
SCC-machine (simplified by observing E)
CK-machine (made the stack to represent E)
↳ still has slow substitution

$M[X \leftarrow V]$ — must look at all of M

Idea: Save pending substitutions and perform when we get to variables normally

$\langle V, fn(\lambda X.M, K)\rangle \longmapsto_{ck} \langle M[X \leftarrow V], K\rangle$
$\langle M, [X \leftarrow V], K\rangle$
new position

Wrong — CEK-machine — Code, Environment, Kontinuation
$st = \langle M, E, K\rangle \quad E = \bullet \mid E[X \leftarrow V]$

$tr(M) = \langle M, \bullet, ret\rangle$

lookup (E, X)

lookup $(E[X \leftarrow V], X) = V$
lookup $(E[Y \leftarrow V], X) =$ lookup (E, X)

1. $\langle X, E, K\rangle \longmapsto_{ck} \langle$ lookup$(E, X), E, K\rangle$
2. $\langle V, E, fn(\lambda X.M, K)\rangle \longmapsto_{ck} \langle M, E[X \leftarrow V], K\rangle$
3. $\langle M\ N, E, K\rangle \longmapsto_{ck} \langle M, E, ar(N, K)\rangle$
4. $\langle V, E, ar(N, K)\rangle \longmapsto_{ck} \langle N, E, fn(V, K)\rangle$

$((\lambda X.\lambda Y.(X+Y))\ 5)\ 6) \xrightarrow{tr} \langle\ ,\bullet, ret\rangle \overset{3}{\mapsto} \langle ④, \bullet, ar(6, ret)\rangle$
$\overset{3}{\mapsto} \langle ③, \bullet, ar(5, ar(6, ret))\rangle \overset{4}{\mapsto} \langle 5, \bullet, fn(③, ar(6, ret))\rangle$
$\overset{3}{\mapsto} \langle ②, \bullet[X \leftarrow 5], ar(6, ret)\rangle \overset{4}{\mapsto} \langle 6, \bullet[X \leftarrow 5], fn(②, ret)\rangle$
$\overset{3}{\mapsto} \langle X+Y, \bullet[X \leftarrow 5][Y \leftarrow 6], ret\rangle \overset{*}{\mapsto} \langle 11, \bullet[X \leftarrow 5][Y \leftarrow 6], ret\rangle$

1-2

$((\lambda X.\lambda Y.\ X+Y)\ 5)\ X)$

$\longmapsto_{CK}\ \langle X, \bullet [X \leftarrow 5], fn(\lambda Y.\ X+Y, ret)\rangle$

$\mapsto\ \langle 5, \bullet [X \leftarrow 5], fn(\lambda Y.\ X+Y, ret)\rangle$

$\mapsto\ \langle X+Y, \bullet [X \leftarrow 5][Y \leftarrow 5], ret\rangle \quad \longmapsto\!\!\!\!\rightarrow\ 10$

$(\lambda F.\ (\lambda X.\ (F\ 6))\ 7))((\lambda X.\ (\lambda Y.\ X))\ 5)$

$(\lambda Y.\ 5)$

$((\lambda X.\ ((\lambda Y.\ 5)\ 6))\ 7)$

$\rightarrow ((\lambda Y.\ 5)\ 6) \rightarrow 5$

Wrong-CEK returns 7.

$\langle X, \bullet [X \leftarrow 5][F \leftarrow (\lambda Y.X)][X \leftarrow 7][Y \leftarrow 6], ret\rangle$

$((\lambda X.\ ((\lambda X.\ X)\ 6))\ 5) \quad \longrightarrow\!\!\!\!\rightarrow\ 6$

$\longmapsto\!\!\!\!\rightarrow_{wrong\text{-}cek^2}\ 5$ (looking at oldest subst)

1. $\langle X, E, K\rangle \longmapsto_{CEK} \langle lookup(E, X), \underbrace{E}_{useless}, K\rangle$

2. $\langle V, E, fn(\lambda X.M, k)\rangle \mapsto \langle M, E[X \leftarrow V], K\rangle$

wrong substitutions

3. $\langle (M\ N), E, K\rangle \mapsto \langle M, E, ar(N, K)\rangle$

4. $\langle V, E, ar(N, k)\rangle \mapsto \langle N, E, fn(V, K)\rangle$

wrong

9-3/

CEK $E = \cdot \mid E[X \leftarrow V]$

$St = \langle M, E, K \rangle$

$M = X \mid \lambda X.M \mid (M\ N) \mid b \mid (o^n\ M \ldots)$

$V = b \mid \sout{\lambda X.M} \mid clo(\lambda X.M, E)$

$K = ret \mid arg(N, E, K) \mid fn(V, K) \mid pr(o, \langle V \ldots \rangle, E, \langle M \ldots \rangle, K)$

$\langle X, E, K \rangle \longmapsto_{CEK} \langle lookup(E,X), \cdot, K \rangle$

$\langle M\ N, E, K \rangle \mapsto \langle M, E, \underline{arg}(N, E, K) \rangle$

$\langle \lambda X.M, E, K \rangle \mapsto \langle clo(\lambda X.M, E), \cdot, K \rangle$

$\langle V, E, \overline{arg}(N, E', K) \rangle \mapsto \langle N, E', \underline{fn}(V, K) \rangle$

$\langle V, E, \overline{fn}(clo(\lambda X.M, E'), K) \rangle \mapsto \langle M, E'[X \leftarrow V], K \rangle$

eval : M → V := eval m = cek m · ret

cek : M × E × K → V :=

cek X E K = cek (lookup E X) · K

cek (M N) E K = cek M E arg(N, E, K)

cek (λX.M) E K = cek clo(λX.M, E) · K

cek V E arg(N, E', K) = cek N E' fn(V, K)

cek V E fn(clo(λX.M, E'), K) = cek M E'[X←V] K

cek V E ret = V

val eval (expr m) {

expr c = m; env e = ·; kont k = ret;

while (1) {

if (c is var) { c = E.lookup(c); e = ·; }

else ...

else (c is val && k is ret) { break; } }

return c; }