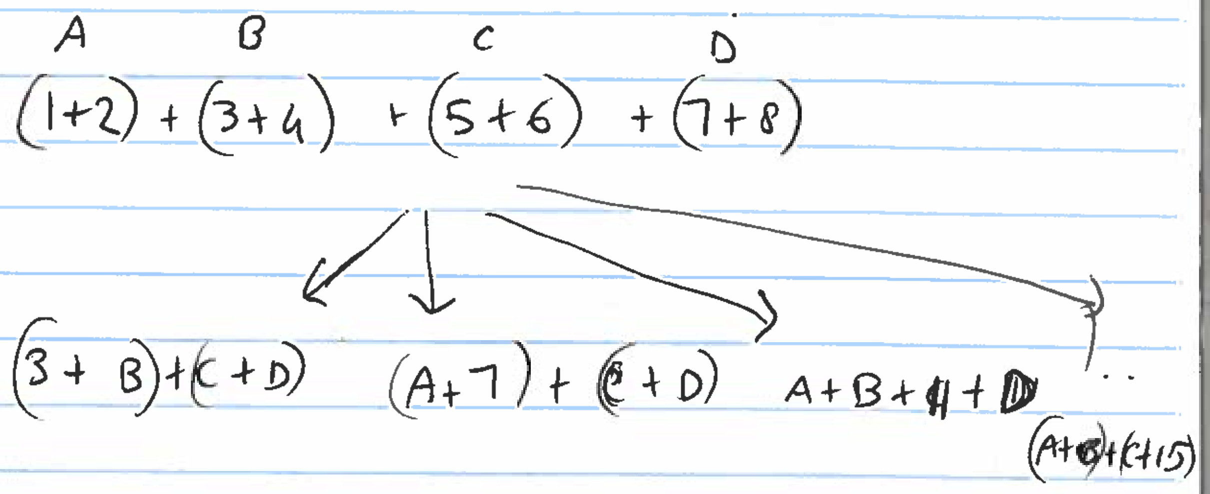
OPL

~~of SRM~~

Last lecture :

- textual reduction machine all contexts are good



non deterministic! bad for implementations

~~From~~ - STANDARD REDUCTION MACHINE.

from Contexts to Evaluation Contexts!

$$E ::= \boxed{} \mid E\ N \mid \underset{\text{must be value!}}{\underline{v}}\ E \mid o^n\ v..\ E\ M_1, M_2..$$

Uniquness of Evaluation Contexts! ∀m

CC machine

Problem with SRM.

Intro:

$E = add_1\ [\,]$ $\quad$ $add_1\ ((\lambda x.(\lambda y.(\lambda z.x\ \ulcorner 3 \urcorner))\ \ulcorner 2 \urcorner)\ \ulcorner 1 \urcorner)$

$e = \lambda x \ldots$

$\quad$ ~~add~~ $add_1(\lambda y\ \lambda z.\ulcorner 1 \urcorner\ \ulcorner 3 \urcorner\ \ulcorner 2 \urcorner)$

$E = add_1 [\,]$

$e = \lambda y \ldots$

$\quad \rightarrow$

$E = add_1\ [\,]$ $\quad$ $add_1(\lambda z.\ulcorner 1 \urcorner\ \ulcorner 3 \urcorner)$

$e = \lambda z.1\ \ulcorner 3 \urcorner$

$\quad \rightarrow add_1\ \ulcorner 1 \urcorner$

$E = ▧$

$e = add_1\ 1$ $\quad \rightarrow\ 2$

E has been recomputed every time!

Also.. a lot hidden...

SRM is good for declarative formalization.

SRM is bad for executing.

Idea: Separate the concept of context an program
at the syntactic level!
Machine <u>exposes</u> the context syntactically.

$$\langle M, E \rangle \longrightarrow \langle M', E' \rangle$$

exp ↗ ↗ ~~eval~~ context

Start $\langle M, [\,] \rangle$ empty context.

Intuitively

$$\langle ((M\ N), P), \square \rangle \longrightarrow \langle (M\ N), (\square\ P) \rangle$$

current focus

current focus — must remember the shape I encountered.

implements the leftmost search for a redex!

$$(M\ N_1\ N_2\ N_3\ N_4 \cdots N_K), \square \rightarrow \langle M\ N_1\ N_2 \ldots N_{K-1}, (\square\ N_K) \rangle$$

$$\rightarrow \langle M\ N_1\ N_2 \ldots N_{K-2}, ((\square\ N_{K-1})\ N_K) \rangle$$

too quick here, explain $E[(\square\ N_{K-1})]$, i.e plugging ~~ctx~~ in a CTX!

until you get to the last $\rightarrow \langle M\ N_1, ((\square\ N_2)\ N_3 \ldots N_K) \rangle$

application

$M\ N$ <

- M is application. well it was one of the N....
- $\lambda x.M.\ V$: V is a value $\langle \lambda x.M\ V, E \rangle \rightarrow \langle M[V/x], E \rangle$
- $\lambda x.M\ N$: N not a value $\langle V\ N, E \rangle \longrightarrow \langle N, V\ E[V \square] \rangle$

program. Consequently the next state of the machine must have $(K\ L)$ as the control string component and must refine the current context E to a context of the shape $E[([\,]\ N)]$. In other words, the CC machine must have a state transition of the form

$$\langle (M\ N), E\rangle \longmapsto_{\mathsf{v}} \langle M, E[([\,]\ N)]\rangle \quad \text{if} \quad M \notin V$$

Search rules for other shapes of applications are needed, too.

After the control string becomes a redex, the CC machine must simulate the actions of the textual machine on a redex. Thus it must have the following two instructions:

$$\begin{array}{lcll}
\langle (o^m\ b_1 \ldots\ b_m), E\rangle & \longmapsto_{\mathsf{v}} & \langle V, E\rangle \ \text{where}\ V = \delta(o^m, b_1, \ldots, b_m) & [\mathsf{cc}\delta] \\
\langle ((\lambda X.M)V), E\rangle & \longmapsto_{\mathsf{v}} & \langle M[X \leftarrow V], E\rangle & [\mathsf{cc}\beta_{\mathsf{v}}]
\end{array}$$

The result of such a transition can be a state that pairs a value with an evaluation contexts. In the textual machine, this corresponds to a step of the form $E[L] \longmapsto_{\mathsf{v}} E[V]$. The textual machine would actually divide $E[V]$ into a new redex L' and a new evaluation context E' that is distinct from E. But the textual machine clearly does not pick random pieces from the evaluation context to form the next redex. If $E = E^*[((\lambda X.M)\ [\,])]$, then $E' = E^*$ and $L' = ((\lambda X.M)\ V)$. That is, the new redex is formed of the innermost application in E, and E' is the rest of E.

Thus, for a faithful simulation of the textual machine, the CC machine needs a set of instructions that exploit the information surrounding the hole of the context when the control string is a value. In the running example, the CC machine would have to make a transition from

$$\langle V, E^*[((\lambda X.M)\ [\,])]\rangle$$

to

$$\langle ((\lambda X.M)\ V), E^*\rangle$$

At this point, the control string is an application again, and the search process can start over.

Putting it all together, the evaluation process on a CC machine consists of shifting pieces of the control string into the evaluation context such that the control string becomes a redex. Once the control string has turned into a redex, an ordinary contraction occurs. If the result is a value, the machine must shift pieces of the evaluation context back to the control string.

The $\longmapsto_{\mathsf{cc}}$ reduction relation on CC machine states is defined as follows:

$$\begin{array}{lcll}
\langle (M\ N), E\rangle & \longmapsto_{\mathsf{cc}} & \langle M, E[([\,]\ N)]\rangle & [\mathsf{cc1}] \\
\quad \text{if } M \notin V & & & \\
\langle (V_1\ M), E\rangle & \longmapsto_{\mathsf{cc}} & \langle M, E[(V_1\ [\,])]\rangle & [\mathsf{cc2}] \\
\quad \text{if } M \notin V & & & \\
\langle (o^n\ V_1 \ldots V_i\ M\ N \ldots), E\rangle & \longmapsto_{\mathsf{cc}} & \langle M, E[(o^n\ V_1 \ldots V_i\ [\,]\ N \ldots)]\rangle & [\mathsf{cc3}] \\
\quad \text{if } M \notin V & & & \\
\langle ((\lambda X.M)\ V), E\rangle & \longmapsto_{\mathsf{cc}} & \langle M[X \leftarrow V], E\rangle & [\mathsf{cc}\beta_{\mathsf{v}}] \\
\langle (o^m\ b_1 \ldots\ b_m), E\rangle & \longmapsto_{\mathsf{cc}} & \langle V, E\rangle & [\mathsf{cc}\delta] \\
 & & \quad \text{where } V = \delta(o^m, b_1, \ldots b_m) & \\
\langle V, E[(U\ [\,])]\rangle & \longmapsto_{\mathsf{cc}} & \langle (U\ V), E\rangle & [\mathsf{cc4}] \\
\langle V, E[([\,]\ N)]\rangle & \longmapsto_{\mathsf{cc}} & \langle (V\ N), E\rangle & [\mathsf{cc5}] \\
\langle V, E[(o^n\ V_1 \ldots V_i\ [\,]\ N \ldots)]\rangle & \longmapsto_{\mathsf{cc}} & \langle (o^n\ V_1 \ldots V_i\ V\ N \ldots), E\rangle & [\mathsf{cc6}]
\end{array}$$

$$eval_{\mathsf{cc}}(M) = \begin{cases} b & \text{if } \langle M, [\,]\rangle \longmapsto\!\!\!\!\rightarrow_{\mathsf{cc}} \langle b, [\,]\rangle \\ \texttt{function} & \text{if } \langle M, [\,]\rangle \longmapsto\!\!\!\!\rightarrow_{\mathsf{cc}} \langle \lambda X.N, [\,]\rangle \end{cases}$$

By the derivation of the CC machine, it is almost obvious that it faithfully implements the textual machine. Since evaluation on both machines is defined as a partial function from

Example CC machine

$((M\,N)\,P)$ $\quad \langle ((\lambda f.\lambda x.(f\ x)\ \ \lambda y.(+\ y\ y))\ \ 1), \blacksquare \rangle$

CC1 $\quad \Downarrow$

$(M\,N) \approx (V_1 V_2)$ $\quad \langle (\lambda f.\lambda x.(f\ x)\ \ \lambda y.(+\ y\ y)), (\blacksquare\ 1) \rangle$

$\Downarrow$

$CC\beta_v$

$\lambda x.M$ $\quad \langle \lambda x.((\lambda y.+\ y\ y)\ \ x), (\blacksquare\ \ 1) \rangle$

ccS $\quad \Downarrow$

$(M\,N) \approx V_1 V_2$ $\quad \langle \lambda x.((\lambda y.+\ y\ y)\ x)\ \ 1\ , \blacksquare \rangle$

$CC\beta_v$ $\quad \Downarrow$

$M N \approx V_1 V_2$ $\quad \langle \lambda y.+\ y\ y\ \ 1, \blacksquare \rangle$

$\Downarrow$

$cc\beta_v$

$op\ V_1\ V_2$ $\quad \langle +\ 1\ 1, \blacksquare \rangle$

$\Downarrow$

$cc\delta$

$\langle 2, \blacksquare \rangle$

Example CC machine

M N $\quad \langle (\lambda f.\lambda x.(f\ x)\quad \lambda y.(+\ y\ y))\ (+\ (\lambda x.x\ 3)\ 2),$ ▨ $\rangle$

$\downarrow$ cc1

$\langle \lambda f\ \lambda x\ (f\ x)\quad \lambda y\ (+\ y\ y),\ ($▨ $(+\ (\lambda x.x\ 3)\ 2)\ )$

$\downarrow$ ccβv

$\lambda x.$ M $\quad \langle (\lambda x.(\lambda y\ +\ y\ y)\ \cancel{y}\ ($▨$(+\ (\lambda x\ x\ 3)\ 2))\rangle$

$\downarrow$ cc5

M M $\quad \langle (\lambda x.(\lambda y\ +\ y\ y)\ x)\ (+\ (\lambda x\ x\ 3)\ 3),$ ▨ $\rangle$

$\downarrow$ cc2

o^2 M N $\quad \langle +\ (\lambda x\ x\ 3)\ 2\quad ,\ (\lambda x.\lambda y(+\ y\ y)\ x$ ▨ $))\rangle$

$\downarrow$ cc3

$V_1\ V_2 \quad \langle (\lambda x\ x\ 3)\quad ,((\lambda x.(\lambda y\ +\ y\ y)\ x)\ (+$ ▨ $2))\rangle$

$\downarrow$ ccβv

value $\quad 3\ ,\quad \ldots$

$\downarrow$ cc6

$\langle (+\ 3\ 2)\ ,\quad (\lambda x(\lambda y\ +\ y\ y)x$ ▨ $)\rangle$

$\downarrow$

$5 \quad \ldots$

$\downarrow$

$(\lambda x.\ \ldots\ 5)$

Example with omega in CC

MN with N not a value $\quad \langle \lambda x.0\ ((\lambda x.xx)\ (\lambda x.xx)),\ ▩ \rangle$

$\downarrow$

cc2

$M_1 V_2$ $\quad \langle (\lambda x.xx)(\lambda x.xx),\ (\lambda x.0\ ▩) \rangle$

$\downarrow$

$cc\beta_v$

$V_1 V_2$ $\quad \langle \lambda x.x\ \ \lambda x.xx,\ \lambda x.0\ ▩ \rangle$

$\downarrow$

$cc\beta_v$

$\langle \lambda x.xx\ \ \lambda xxx,\ \lambda x.0\ ▩ \rangle$

$\vdots$

Problem with CC:
it only exploit the info
in ~~Current focus~~ control string

Drawback 1 of CC: if I know I have a function in the context and I get a value

$\langle \lambda x.x\ 5,\ \lambda x.x\ \blacksquare \rangle$

↓

$\langle 5,\ (\lambda x.x\ \blacksquare) \rangle$

↓

$(\lambda x.x\ 5),\ \blacksquare$

silly...
we can perform the computation, if we look at the contexts!

$(\lambda x.x\ 5)$

↓

$\langle 5,\ \blacksquare \rangle$

Drawback 2 of CC:

$\langle (\lambda x.x\ \ \lambda x.x)\ (\lambda x.x\ \ulcorner 5 \urcorner),\ \blacksquare \rangle$

↓

$\langle \lambda x.x\ \ \lambda x.x,\ (\blacksquare\ \ \lambda x.x\ \ulcorner 5 \urcorner) \rangle$

↓

$\langle \lambda x.x,\ (\blacksquare\ \ (\lambda x.x\ \ulcorner 5 \urcorner)) \rangle$

↓

$\langle \lambda x.x\ \ (\lambda x.x\ \ulcorner 5 \urcorner),\ \blacksquare \rangle$

↓

$\langle (\lambda x.x\ \ulcorner 5 \urcorner),\ (\lambda x.x\ \blacksquare) \rangle$

silly because CC must put the function in CTX anyway, you can do the swap directly

$\langle (\lambda x.x\ \ulcorner 5 \urcorner)\ \ (\lambda x.x\ \blacksquare) \rangle$

CC has 3 rules for application disjointly and only side-conditions keep them distinctly used.

MN {
- M is not a value
- M is a value but N is not
- M and N are values.

SCC = 1 rule for application
no side conditions

Example SCC machine

M N < λf λx(f x) (λy +yy) (+ (λx.x 3) 2), ▨ >

scc1 ↓

V₁V₂ < < λf.λx(f x) (λy(+yy)) , (▨ (+ (λx x 3) 2)) >

scc1 ↓

< λf.λx(f x) , ((▨ (λy + yy))(+ (λx.x 3) 2)) >

scc3 ↓

~~< λx.(λy+yy) , (▨ (+ (λx x 3) 2)) >~~

scc4 (SWAP)

Vm λy + yy   (λf λx f x ▨) +(λx(x 3) 2)
Evil Th value

sec3 ↓

< λx. (λy+yy) x , (▨ (+ λx(x3) 2)) >

scc4 ↓

+ (λx x3) 2 , λx λy +yy ▨

scc2 ↓

(λx.x 3) , (λx. λy(+ y y)) (+ ▨ 2)

scc1 ⇓

λx.x λx. λy+yy (+ (▨ 3) 2)

scc4 ⇓

3 λx. λy +yy (+ (λx.x ▨) 2)

scc3 ↓

3 λx λy+yy (+ ▨ 2)

scc6 ↓

2 , λx λy+yy (+ 3 ▨)

scc5 ↓

5, λ λy+yy ▨

scc3 ↓

+55 , ▨ → 5, +45 → 5, +5 ▨

↓

10, ▨