Lecture 23

Recall Call-by-name
λ-calculus

Lazy Functional Languages
(Haskell, Miranda)

motivation $(\lambda x.(\lambda y.\, y))\ \Omega = \lambda y.y$

and not an ∞-computation

- "Referential Transparency".

In terms of the λ-calculus

$$(\beta)\quad ((\lambda x.e_1)\ e_2) \rightarrow_\beta e_1[e_2/x]$$

$$e \longrightarrow e'$$

$$(\text{op}) \qquad \frac{e_1 \longrightarrow_\beta e_1'}{e_1\, e_2 \longrightarrow_\beta e_1'\, e_2}$$

$$(\text{arg}) \qquad \cancel{\frac{e_2 \longrightarrow_\beta e_2'}{e_1\, e_2 \longrightarrow_\beta e_1\, e_2'}}$$

$$(\xi) \qquad \cancel{\frac{e \rightarrow_\beta e'}{\lambda x.e \longrightarrow_\beta \lambda x.e'}}$$

Note: CBN does not use (arg) and (ξ) rules.

In terms of Contexts and Reduction

$$
\begin{aligned}
\mathcal{C}[\;] ::= \;& [\;] \\
\mid\;& \mathcal{C}[\;]\; e \\
\mid\;& \cancel{e\; \mathcal{C}[\;]}
\end{aligned}
$$

$\cancel{\mid\ e\ \mathcal{C}[\ ]}$

$\cancel{\mid\ \lambda x.\,\mathcal{C}[\ ]}$

CBN does not use contexts $e\ \mathcal{C}[\ ]$ (arg), and $\lambda x.\,\mathcal{C}[\ ]$ (ξ)

$$\mathcal{C}[(\lambda x.e_1)\ e_2] \to_\beta \mathcal{C}[e_1[e_2/x]]$$

where

$$\mathcal{C}[\ ] ::= [\ ]$$
$$\mid\ \mathcal{C}[\ ]\ e$$

is a CBN Evaluation Context.

Any evaluation context $\mathcal{C}[\ ]$ can be represented as a <u>stack</u> of Basic Contexts

$$B[\ ] ::= [\ ]\ \mid\ [\ ]\ e$$

Build $\mathcal{C}[\ ]$ by placing

the B[ ] as "Russian Matryona Dolls".

Evaluation Strategy

- Find a redex in left "spine" (left-most, outermost)
- Log the context C[ ]
- Reduce the redex, keeping context intact
- ☆ If no redex found, we are in cbn-nf. (WHNF)

Weak Head Normal Form

$e ::= x \mid \lambda x.e \mid$

$x\, e_1 \cdots e_n \quad (n \geq 1)$

$x\, \underbrace{e_1 \cdots e_n}_{\text{(unevaluated)}} \quad (n \geq 0)$

CONTEXT-REDUCTION "MACHINE"

MACHINE in the sense that it is an algorithm ... rewriting a machine configuration one step at a time

Configurations:

$$\text{Exp} \times (\text{Stack of basic contexts})$$

$$\langle e, \boxed{S} \rangle$$

focus - expression

can be "squished"

to yield a context
$e[\ ]$

- "stack up" basic contexts

$$B[\ ] ::= [\ ] \quad \text{— hole}$$
$$\quad\quad\quad \mid\ [\ ]\, e$$

hole in op-position

argument

(Decompose) — Looking for redex

$$\langle e_1\ e_2,\ \boxed{S} \rangle \Longrightarrow \langle e_1,\ \boxed{\begin{matrix}[\ ]\ e_2 \\ S\end{matrix}} \rangle$$

": Shift focus to op-position e_1

- push context $[\ ]\, e_2$ onto stack

$$\mathcal{C}[\, e_1\ e_2\,] \dashrightarrow \mathcal{C}[\, \underline{[e_1]\ e_2}\,]$$

(apply) – found a redex, now reduce.

$$\left\langle \lambda x. e_1 ,\ \begin{array}{|c|} [\ ]\, e_2 \\ S \\ \hline \end{array} \right\rangle \Longrightarrow$$

$$\left\langle e_1[e_2/x] ,\ \begin{array}{|c|} S \\ \hline \end{array} \right\rangle$$

- Pop the basic context $[\ ]\, e_2$ off stack
- Replace focus by "contractum"
  $e_1[e_2/x]$

$$\mathcal{C}[\, [\lambda x. e_1]\ e_2\,] \to \mathcal{C}[\, e_1\, [e_2/x]\,]$$

Note: — the "decompose" step is only analysing the context, and

— "apply" is doing the real computation

"COLLAPSING" a Stack of Basic Contexts

$$B[\ ] ::= [\ ] \mid [\ ]\,e$$

into an evaluation context $\mathcal{E}[\ ]$

Let $\mathrm{coll}\left(\ \sqcup\ \right) = [\quad]$

empty stack

context that is only a "hole"

$$\mathrm{coll}\left(\begin{array}{|c|} [\ ]\,e \\ S \\ \hline \end{array}\right) =$$

let $\mathcal{E}[\quad] = \mathrm{coll}(|S|)$

in

$$\mathcal{C}[\,[\ ]\,e\,]$$

Now, at any point in the execution the expression corresponding to machine configuration

$$\langle\, e,\ \lfloor S \rfloor \,\rangle$$

represents the expression $\mathcal{C}[\ e\ ]$

where $\mathcal{C}[\quad] = \text{coll}(\lfloor S \rfloor)$

NOTE

Since stacked basic closures are always of the form $[\ ]\,e$, we can just

stack up E, leaving the [illegible]
position hole [ ] implicit.

CLOSURES
&
CLOSURE MACHINES

We know the $e_1[e_2/x]$ is an expensive operation...

So we build "closures"

- expression (with variables)

+

- table (to find bindings for "apparently free" but actually bound variables, which

have to be
substituted for)

Data structure:

$$\text{Closure} \triangleq \text{Exp} \times \text{Table}$$

and

$$\gamma \in \text{Table} \triangleq X \xrightarrow[\text{fin}]{} \text{Closure}$$

(mutually recursive)

Well-founded, since $e \in \text{Exp}$
— need not have a free variable
— and $\gamma \in$ Table can be empty

VClosure

$$\boxed{\text{Let } val ::= (\lambda x.\overset{\text{value}}{e}, \overset{\text{table}}{\gamma})}$$

$$\mid \underbrace{(x\ e_1 \ldots e_n, \gamma)}_{\text{if you really insist}}$$

Closure-Semantics for CBN evaluation

$\Longrightarrow_n \subseteq \text{Closure} \times \text{VClosure}$

Input - Output Spec

$$(\text{var})\quad \frac{\gamma(x) \Longrightarrow_n \text{val}}{(x, \gamma) \Longrightarrow_n \text{val}}$$

$$(\text{abs})\quad \frac{}{(\lambda x.e, \gamma) \Longrightarrow_n (\lambda x.e, \gamma)}$$

$$(\text{app})\quad \frac{(e_1, \gamma) \Longrightarrow_n (\lambda x.e', \gamma') \qquad (e', \gamma'[x \mapsto (e_2, \gamma)]) \Longrightarrow_n \text{val}}{(e_1\ e_2, \gamma) \Longrightarrow_n \text{val}}$$

(var): • If $x \in \text{dom}(\gamma)$, look up the table for the closure corresponding to x and evaluate it to an answer. (val)

(abs) • An abstraction closure is an answer already (0 steps) (app) · CBN function call . $e_1\ e_2$

- First simplify the closure (e_1, γ) corresponding to op e_1 to an abstraction closure $(\lambda x.e', \gamma')$
- Then evaluate the body e' wrt γ' extended by binding formal parameter x to the argument closure (e_2, γ)
- Return the result val

COMBINING THE CLOSURE REPRESENTATION WITH THE CONTEXT M/C

(V)

$$\langle (x, \gamma), \boxed{S} \rangle \Longrightarrow$$

$$\langle \gamma(x), | S | \rangle$$

- Look up variable x in table γ, and now focus on that closure for further reduction.

(β)

$$\left\langle (\lambda x.e_1 , \gamma) , \boxed{\begin{matrix}(e_2 , \gamma') \\ S\end{matrix}} \right\rangle \Longrightarrow \left\langle (e_1 , \gamma[x \mapsto (e_2 , \gamma')]) , \boxed{S} \right\rangle$$

Focus now on <u>body</u> e_1 of the abstraction extending the table γ by binding formal parameter x to the argument closure (e_2 , γ') which is popped off the stack.

(first step of a function call)

(D)

$$\left\langle ((e_1\ e_2), \gamma),\ \boxed{S} \right\rangle \Longrightarrow \left\langle (e_1, \gamma),\ \boxed{\begin{matrix}(e_2,\gamma)\\ S\end{matrix}} \right\rangle$$

Shift focus from $e_1\ e_2$ to op-position e_1, pushing arg closure (e_2, γ) on stack

---

When does this machine (KRIVINE Machine) stop?

— when we have a config

$- \langle (x, \gamma), \sqcup \rangle$

but $x \notin dom(\gamma)$.

OR

$- \langle (\lambda x. e, \gamma), \underbrace{\sqcup}_{\text{empty stack.}} \rangle$

i.e when "unloading" the machine gives an expression in <u>Weak HNF.</u>

<u>Call-by-value</u>

(β_v) $\overline{\quad (\lambda x. e)\, v \to e[v/x] \quad}$

(β_v) $\qquad (\lambda x.e_1)\ v \longrightarrow_\beta e_1[v/x]$

v — a value.

$$(op) \qquad \frac{e_1 \longrightarrow_\beta e_1'}{e_1\ e_2 \longrightarrow_\beta e_1'\ e_2}$$

$$(arg_v) \qquad \frac{e_2 \longrightarrow_\beta e'}{v\ e_2 \longrightarrow_\beta v\ e'}$$

(No (ξ) rule)

Evaluation Contexts for CBV

$$\mathcal{E}[\ ] ::= [\ ] \mid \mathcal{E}[\ ]\ e \mid v\ \mathcal{E}[\ ]$$

$$v ::= \lambda x.e \mid x\ v_1 \ldots v_n$$

if you really insist...

Context Search is for left-most inner-most (not below λ) redex.

Context Machine for CBV.

Basic Contexts:

$$B_v[\ ] ::= [\ ]\,e \mid v\,[\ ] \mid [\ ]$$

Any evaluation context $\mathcal{E}[\ ]$ is obtained by "collapsing in" a stack of Basic Contexts.

Machine Configurations:

$$\langle e, \boxed{S} \rangle$$

focus expression

stack of basic contexts

Note: $\underbrace{\boxed{}}_{\text{empty stack}} = \underbrace{[\ \ ]}_{\text{hole}}$

(Decompose)

$$\langle e_1\ e_2, \boxed{S} \rangle \Longrightarrow$$

$$\langle e_1, \boxed{\begin{array}{c}[\ \ ]\ e_2 \\ S\end{array}} \rangle$$

As before

(Arg)

$$\left\langle \lambda x.e_1 , \boxed{\begin{array}{c}[\ ]\,e_2\\ S\end{array}} \right\rangle \Rightarrow_1 \left\langle e_2 , \boxed{\begin{array}{c}(\lambda x.e_1)[\ ]\\ S\end{array}} \right\rangle$$

Note: Having found an abstraction in the op-position, swap contexts on the top-of-stack, to now focus on the arg position

$$\mathcal{C}[\,[\lambda x.e_1]\,e_2\,] \dashrightarrow \mathcal{C}[\,\lambda x.e_1\,[e_2]\,]$$

(Apply)

$$\left\langle v, \boxed{\begin{array}{c}(\lambda x.e_1)[\ \ ]\\ S\end{array}} \right\rangle \Longrightarrow$$

$$\left\langle e_1[v/x], \boxed{S} \right\rangle$$

$$\mathcal{C}\big[(\lambda x.e_1)[v]\big] \rightarrow_{\beta v} \mathcal{C}\big[e_1[v/x]\big]$$

Again: at any point in the execution, configuration

$$\left\langle e, \boxed{S} \right\rangle$$

corresponds to expression

$$\mathcal{E}[\,e\,] \quad \text{where}$$

$$\mathcal{E}[\quad] = \mathrm{coll}\left(\boxed{S}\right)$$

(for a slightly extended definition of coll)

Exercise: Extend the definition of coll.

Closures for call-by-value

$$Table \triangleq X \to_{fin} VClosure$$

$$vcl \in VClosure \triangleq \{ (\lambda x.e, \gamma) \mid e \in Exp,$$

and $\gamma \in$ Table }

Cbv rules with closures

$$(\text{var})\ \frac{}{(x, \gamma) \Rightarrow_v \gamma(x)}$$

a value closure

$$(\text{abs})\ \frac{}{(\lambda x.e, \gamma) \Rightarrow_v (\lambda x.e, \gamma)}$$

$$(\text{app})\ \frac{(e_1, \gamma) \Rightarrow_v (\lambda x.e', \gamma') \quad (e_2, \gamma) \Rightarrow_v val_2 \quad (e', \gamma'[x \mapsto val_2]) \Rightarrow_v val_3}{(e_1\, e_2, \gamma) \Rightarrow_v val_3}$$

(var) · Look up table γ for x – return the value closure $\gamma(x)$

(abs) - Already a value closure

(app) - First evaluate e_1 wrt γ to

(app)

value closure $(\lambda x. e', \gamma')$

- Then evaluate e_2 wrt γ to value closure vcl_2
- Finally evaluate body of abstraction, i.e, e' wrt γ' extended by x bound to vcl_2
- and return the result vcl_3