CSCI 7000-11

Substitution : $M[N/x]$ Subst : $\overset{M}{Exp} \to \overset{N}{Exp} \to \overset{x}{Strg} \to Exp$

Definition 1 :

$$x[s/x] = s$$

$$y[s/x] = y \quad \text{if } x \neq y$$

$$(\lambda y.\, t_1)[s/x] = \lambda y.\, t_1[s/x]$$

$$(t_1\ t_2)[s/x] = t_1[s/x]\ t_2[s/x]$$

$$(\lambda x.\, x)\left[y/x\right] \neq (\lambda x.\, y)$$

Counter Example

Bound

Id ——— Constant.

Definition 2 :

$$x[s/x] = s$$

$$y[s/x] = y \quad \text{if } x \neq y$$

$$(\lambda y.\, t_1)[s/x] = \lambda y.\, t_1[s/x] \quad \text{if } x \neq y$$

$$(t_1\ t_2)[s/x] = t_1[s/x]\ t_2[s/x]$$

$$\underbrace{(\lambda x.\, y)}_{\text{Constant}}\left[\overset{s}{x\ x}/y\right] \longrightarrow \underbrace{(\lambda x.\, x\ x)}_{\Omega}$$

Capturing Substitution.

"Capture-Avoiding" Substitution:

Definitions :

$x[s/x] = s$

$y[s/x] = y$ if $x \neq y$

$(\lambda y. t_1)[s/x] = \lambda y. t_1[s/x]$ if $x \neq y \wedge y \notin FV(s)$

$(t_1\ t_2)[s/x] = t_1[s/x]\ t_2[s/x]$

ω is free

$(\lambda x. x\,y)\left[\overset{s}{x\,x}/y\right] = \lambda \omega. (\omega/x]\, x\, y\left[x\,x/y\right]$

α-Equivalence

$\lambda \omega. [x\,x/y][\omega/x](x\ y)$

$(\omega\ y)$

$= \lambda \omega.\ \omega\ x\, x$

Definition 4 :

$$x[s/x] = s$$

$$y[s/x] = y \quad \text{if } x \neq y$$

$$(\lambda y.\, t_1)[s/x] = \lambda y.\, t_1[s/x] \quad \text{if } x \neq y \wedge y \notin FV(s)$$

$$(\lambda y.\, t_1)[s/x] = \lambda w.\, t_1\left[\tfrac{w}{y}\right]\left[\tfrac{s}{x}\right] \text{ if } x \neq y \wedge y \in FV(s)$$

w is fun

$$w \notin FV(s) \cup F(t_1) \cup \{x, y\}$$

$$(t_1\ t_2)[s/x] = t_1[s/x]\ t_2[s/x]$$

FV : Exp → String set

(Var " ")

(Abs)

(App)

$FV(x) = \{x\}$

$FV(y) = \{y\}$

→ Reduction Rules

$$e ::= x \mid (\lambda x.\, e) \mid e_1\ e_2$$

1) Variable $\quad x \quad$ (Already in Normal form)

2) Lambda Abstraction

$$\frac{e \rightarrow e'}{(\lambda x.\, e) \rightarrow (\lambda x.\, e')} \quad [E\text{-}Abs]$$

3) Application:

"Congruence" rules

$$\frac{e_1 \longrightarrow e_1'}{e_1\ e_2 \longrightarrow e_1'\ e_2}\quad [\text{E-App-1}]$$

$$\frac{e_2 \longrightarrow e_2'}{e_1\ e_2 \longrightarrow e_1\ e_2'}\quad [\text{E-App-2}]$$

"Reduction" rule

$$\frac{}{(\lambda x.e_1)\ e_2 \longrightarrow [e_2/x]\ e_1}\quad [\text{E-App-3}]$$

$$= e_1\ [e_2/x]$$

Full-Beta reduction.

2/17

CSCI 7000 - 11

Today's lecture:

* Recap of dynamic semantics
  * Full-beta reduction ($\to_\beta$)
* Non-deterministic evaluation
  * Example: $(\lambda x.\ x\ x)((\lambda x.y)\ z)$
* β-normal form (no β redexes)
* Multi-step reduction
  * Reflexive transitive closure of ($\to_\beta$)
* Church-Rosser Theorem
* Lack of normal forms: Ω combinator
  ↳ contradiction?
* Reduction strategy: Normal order vs
  Call-by-name/need vs
  Call-by-value

Example: $(\lambda x.x)((\lambda x.x)(\lambda z.(\lambda x.x)z))$

CBN Example: $(\lambda x.(x\ y)(x\ z))((\lambda x.x)\ a)$

$$FV(\underbrace{(\lambda x.\ x\ x)}_{e_1}\underbrace{((\lambda x.y)\ z)}_{e_2}) = \{y, z\}$$

$$\to_\beta ((\lambda x.y)\ z)\ ((\lambda x.y)\ z)$$

$y \quad ((\lambda x.y)\ z)$

$\to_\beta (y\ y)$ } App(Var "y", Var "y")

$\{y, z\}$: y → one ↓ $\lambda f.\lambda x.(f\ x)$; → z

$$(\lambda x.\, x\, x)\,((\lambda x.\, y)\, z)$$

$$(\lambda x.\, x\, x)\quad y$$

$$(y\; y)$$

$$(\lambda x.\, e_1)\; e_2$$

$$e_1[e_2/x]$$

Church Rosser:

$$\begin{tikzcd}
 & M' \arrow[dr, "\beta^*"] & \\
M \arrow[ur, "\beta^*"] \arrow[dr, "\beta^*"'] & & N \\
 & M'' \arrow[ur, "\beta^*"'] &
\end{tikzcd}$$

$$(\lambda x.\, x\, x)\;(\lambda x.\, x\, x)$$

Ω Combinator

N

$$(\lambda x.\, x\, x)\;(\lambda x.\, x\, x)$$

Normal order reduction

"Left-most outer-most redex first"

$$(\lambda x.\, x)\;((\lambda x.\, x)\;(\lambda z.\, (\lambda x.\, x)\, z))$$

$$\xrightarrow{\;BN\;} \quad Id\;(Id\;(\lambda z.\, Id\; z))$$

$$=$$

$$Id\;(\lambda z.\, Id\; z)$$

$$\underbrace{(\lambda z.\ Id\ z)}$$

$$(\lambda z.\ \underbrace{z}_{Id})$$

Call-by-Value

→ Evaluate arguments first

→ Do not evaluate inside lambda Abstraction.

[fun c1 → raise (Invalid Arg...)]

Eager evaluation

Strict evaluation

$$(\lambda x.x)\ ((\lambda x.x)\ (\lambda z.\ (\lambda x.x)\ z))$$

$$\underline{Id}\ (\underbrace{\overset{e_1}{Id}\ (\overbrace{\lambda z.\ Id\ z}^{e_{11}})}_{})$$

$\xrightarrow{\beta v}$ e_1 ↓ e_2

$$Id\ \underbrace{(\lambda z.\ Id\ z)}$$

↓

$$(\lambda z.\ \underbrace{Id\ z}_{z})$$

Expensive

$$\underline{(\lambda x.\ y)\ (f)}$$

- factorial 1000

→ Call-by-Name Semantics

Lazy

non-strict

→ Evaluate top-level ~~β~~-redexes first

→ Do not evaluate inside lambda Abstraction.

$$(\lambda x. x)\ ((\lambda x. x)\ (\lambda z. (\lambda x. x) z))$$

$$\underbrace{Id}_{e_1}\ (\underbrace{Id\ (\lambda z.\ Id\ z)}_{e_2})$$

$\longrightarrow_{\beta N\alpha}$

$$\underbrace{Id}_{e_1}\ (\lambda z.\ Id\ z)$$

$$(\lambda z.\ Id\ z)$$

$$(\lambda x.\ y)\ (\underbrace{factorial\ 10^3}) \quad \text{Avoid this case!}$$

$$\downarrow$$

$$(\lambda x.\ \underline{x + x})\ (\underbrace{factorial\ 10^3}_{10^{99}})$$

$x + x$

call-by-need)

$$:\ (\underbrace{\lambda x. (x\ y)\ (x\ z)}_{f})\ (\underbrace{(\lambda x. x)\ a}_{arg})$$

$$(\underbrace{(\lambda x. x)\ a}\ y)\quad ((\underbrace{\lambda x. x)\ a}\ z)$$

* α-renaming $\qquad e_1 =_\alpha e_2$

* β-reduction $\qquad e_1 \longrightarrow^*_\beta e_2$

$e_1 =_\beta e_2$

* η-reduction $\qquad e_1 =_\eta e_2$

η- Elimination

$(\underbrace{\text{fun } x \to \text{foo } x}) \qquad (\lambda x.\ \text{foo } x)$

$=_\eta \text{foo}$

$\underbrace{[1;\ 2;\ 3]} \quad 3\ell$

$\downarrow$

["1"; "2"; "3"]

List.map $(\underline{\text{fun } x \to \text{string\_of\_int } x})$ ℓ

List.map string_of_int ℓ

Church Encoding

* numbers ✓
* Booleans :

: true $\qquad (\lambda \overbrace{x.\ \lambda y.\ x}^{\text{true}})$

: false $\qquad (\underbrace{\lambda x.\ \lambda y.\ y})$

ite $= (\underbrace{\lambda b.\ \lambda x.\ \lambda y.\ b\ x\ y})$ false (ite) true e_1 e_2 $\longrightarrow$ e_1

ite false e_1 e_2 $\longrightarrow$ e_2

$\rightarrow$ $FN =_\beta N$

$YF = F(YF)$

$\rightarrow$ $Y = \lambda f.\ (\lambda x.\ f(x\ x))\ (\lambda x.\ f(x\ x))$

$YF = (\lambda x.\ F(x\ x))\ (\lambda x.\ F(x\ x))$

$F(\lambda x.\ F(x\ x))\ (\lambda x.\ F(x\ x))$

$\rightarrow$ fact : $\lambda f.\ \lambda n.$ if $n = 0$ then 1 else $n * f\ (n-1)$

(Y fact) must compute fixpoint of fact s.t. $\underbrace{(Y\ fact)}_{n}\ n = n!$

$\Rightarrow f(n) = n!$

$FN =_\beta N$

$F(YF) = (YF)$

CBV fp: $Z = \lambda f.\ (\lambda x.\ f\ (\lambda y.\ x\ x\ y))\ (\lambda x.\ f\ (\lambda y.\ x\ x\ y))$

$\rightarrow$ Lists :

nil : $\lambda c.\ \lambda n.\ n$

cons : $\lambda h.\ \lambda t.\ \lambda c.\ \lambda n.\ c\ h\ t$

cons ch H = $\lambda c.\ \lambda n.\ c\ h\ t$

cons ch nil = $\lambda c.\ \lambda n.\ c\ h\ (\lambda c.\ \lambda n.\ n)$

$= \lambda c.\ \lambda n.\ c\ h\ (\lambda c.\ \lambda n.\ n)$

(plus m n   mult n ..

$$\underbrace{(\lambda b.\ \lambda x.\ \lambda y.\ b\ x\ y)}_{\underline{\text{ite}}}\ \text{true}\ e_1\ e_2 \longrightarrow \text{true}\ e_1\ e_2$$

$$\downarrow$$

$$e_1$$

and : $\lambda b\ \lambda c.\ \underline{b}\ c\ \text{fls}$

$(\lambda x.\ \lambda y.\ x)\ \underbrace{(\lambda x\ \lambda y.\ y)}\ \text{fls}$

(plus m n)   (mult n n)

b true c

b fls fls