CS383 Programming Languages

Quiz 4

Quiz: Write the rules for Right-to-Left call-by-value O.S.?

left to right .

(\x.e) v → e [v/x]

$$\frac{e1 \rightarrow e1'}{e1\ v \rightarrow e1'\ v}$$

$$\frac{e2 \rightarrow e2'}{e1\ e2 \rightarrow e1\ e2'}$$

right-to-left call-by-value

Quiz: Evaluate test fls a b?

tru = \t.\f. t   fls = \t.\f. f

test = \x.\then.\else. x then else

(\x.\then.\else. x then else) (\t.\f. f) a b

⇒(\ten.\else.(\t.\f. f) then else) a b

⇒(\t.\f. f) a b

⇒(\f. f) b

⇒b

Quiz: Define succ in lambda calculus

$$\begin{aligned}
\mathbf{0} &= \lambda f.\lambda x.\ x \\
\mathbf{1} &= \lambda f.\lambda x.\ f\ x \\
\mathbf{2} &= \lambda f.\lambda x.\ f\ (f\ x) \\
\mathbf{3} &= \lambda f.\lambda x.\ f\ (f\ (f\ x)) \\
&\ \ \cdots \\
\underline{\mathbf{n}} &= \lambda \underline{f}.\lambda \underline{x}.\ \underline{f^n}\ \boxed{x} \\
&\ \ \cdots
\end{aligned}$$

☆ succ = \n.\f.\x. f (n f x)

succ = \n.\f.\x. n f (f x)

$\text{succ } n = (\lambda \underline{n} f x.\ f\ (\boxed{n}\ f\ x))\ (\lambda g.\lambda y.\ g^n\ y)$

$\to \lambda f.\lambda x.\ f\ \big(\underline{(\lambda g.\lambda y.\ g^n\ y)\ f\ x}\big)$

$\to^* \lambda f.\lambda x.\ f\ f^n\ x$

$= \lambda f.\lambda x\ f^{n+1}\ x = n+1$

Quiz: Why does Γ contain just one instance of (x, t), for any t? In other words, each variable appears only once in Γ.

$$\frac{\Gamma, x : t_1 \mid\!- e_2 : t_2}{\Gamma \mid\!- \lambda x : t_1 . e_2 : t_1 \to t_2}$$

Typing

$[\Gamma \vdash e : t]$

$$\frac{x : t \in \boxed{\Gamma}}{\Gamma \vdash x : t}$$ $x : t$ (T-Var)

$$\frac{}{\Gamma \vdash true : bool}$$ (T-True)

$$\frac{}{\Gamma \vdash false : bool}$$ (T-False)

$$\frac{\Gamma \vdash e_1 : bool \quad \Gamma \vdash e_2 : t \quad \Gamma \vdash e_3 : t}{\Gamma \vdash \text{if } e_1 \text{ then } e_2 \text{ else } e_3 : t}$$ (T-If)

$x : t_1 \quad x : t_2$.

$$\frac{\Gamma\underline{,} x : t_1 \vdash e_2 : t_2}{\Gamma \vdash \lambda x : t_1 . e_2 : t_1 \to t_2}$$ (T-Abs)

$$\frac{\Gamma \vdash e_1 : t_{11} \to t_{12} \qquad \Gamma \vdash e_2 : t_{11}}{\Gamma \vdash e_1\ e_2 : t_{12}}$$ (T-App)

Quiz: Why does Γ contain just one instance of (x, t), for any t? In other words, each variable appears only once in Γ.

$$\frac{\Gamma, x:t_1 \vdash e_2 : t_2}{\Gamma \vdash \lambda x:t_1.e_2 : t_1 \rightarrow t_2}$$

$\Gamma \quad x:t_1 \quad \overset{}{\underset{y}{\cancel{x}}}:t_2 .$

We add binding to Γ in t-abs. For t-abs, we will do alpha-renaming to make sure x is not in Γ.

p101

To avoid confusion between the new binding and any bindings that may already appear in Γ, we require that the name x be chosen so that it is distinct from the variables bound by Γ. Since our convention is that variables bound by λ-abstractions may be renamed whenever convenient, this condition can always be satisfied by renaming the bound variable if necessary. Γ can thus be thought of as a finite function from variables to their types. Following this intuition, we write *dom*(Γ) for the set of variables bound by Γ.

Quiz

church numeral

$C_m = \lambda f. \lambda x. f^m x$

$C_{m+1} = \lambda f. \lambda x. f (f^m x)$.

- $\lambda n. \lambda f. \lambda x. f (n f x)$

succ n $\left(\lambda n. \lambda f. \lambda x\, f (n f x)\right)(\lambda g. \lambda y. g^m y)$

$\rightarrow \lambda f. \lambda x. f((\lambda g. \lambda y. g^m y) f x)$

$\rightarrow \lambda f. \lambda x.\ f\ f^m\ x$

$\rightarrow \lambda f. \lambda x.\ f^{m+1}\ x$.

- $\lambda n. \lambda f. \lambda x. n f (f x)$

succ n $= (\lambda n. \lambda f. \lambda x. n f (f x))\ (\lambda g. \lambda y. g^m y)$

$\rightarrow \lambda f. \lambda x. (\lambda g. \lambda y. g^m y)\ f\ (f\ x)$.

$\rightarrow \lambda f. \lambda x. (\lambda y. f^m\ y)\ (f\ x)$.

$\rightarrow \lambda f. \lambda x.\ f^m\ f\ x$

$= \lambda f. \lambda x.\ f^{m+1}\ x$.

T-abs : definition.

Γ : gamma. context. list (a certain order)

comma operator : add a new binding $(x:t)$ on the right.

$x:t \in \Gamma$: mathmatical definition. no concern about order.

add binding in T-abs : alpha renaming. (rename bound variables)

---

hw4.

note: undefined for natural numbers < 0

succ $0 = 0$.

1. sub m n $\overset{\text{def}}{=}$ m − n.

$\Rightarrow \lambda x. \lambda y.\ y$ pred x. apply y times pred on x.

2. iszero n.

$0 = \lambda f. \lambda x.\ x$ no f inside.

alpha-renaming

$0: (\lambda n. n\ (\lambda x. fls)\ tru)\ (\lambda f. \lambda y. y)$.

$\rightarrow (\lambda f. \lambda y. y)\ (\lambda x. fls)\ tru$.

$\rightarrow^*$ tru. $\lambda t. \lambda f. t$ (first element)

not 0: $(\lambda n. n\ (\lambda x. fls)\ tru)\ (\lambda f. \lambda y. f^m\ y)$

$\rightarrow (\lambda f. \lambda y. f^m y)\ (\lambda x. fls)\ tru$.

$\rightarrow \lambda y. \underbrace{(\lambda x. fls)^m\ y}\ tru$.

$\rightarrow \lambda y.\ fls\ tru$

$\rightarrow fls$. $\lambda t. \lambda f. f$ (second element)

3. leq $= \lambda m. \lambda n.$ iszero (sub m n)

4. equal $= \lambda m. \lambda n.$ and (leq m n) (leq n m).

5. factorial.

by pair: $(\underset{\text{result}}{n!}, \underset{\text{next number}}{n+1})$.

first pair : zz = pair 1 1

"successor" : ss = $\lambda \underset{\text{take one pair as input}}{p}$. pair (multi (fst p) (snd p)) (succ (snd p))

$\Rightarrow$ factorial $= \lambda x.$ fst (x ss zz) ss applied to zz for x times.

- by self-application
  - fact $\overset{def}{=}$ if $(n==0)$ then 1 else $n \cdot \text{fact}(n-1)$.

    $\hookrightarrow \lambda n.(\text{iszero } n)\ 1\ (\text{multi } n\ \underline{\text{fact}}\ (\text{pred } n))$

    tru   fls .   ⚠ recursive .

⇒ self application : apply to itself .

- we want : fact $= \underbrace{(\lambda y.y\ y)\ (\lambda f n.(iszero\ n)\ 1\ (times\ n\ (f\ f\ (pred\ n))))}$ answer .

$\rightarrow \lambda n.(\text{iszero } n)\ 1\ (\text{multi } n\ \text{fact}'\ \text{fact}'\ (\text{pred } n))$

$\rightarrow \lambda n.(\text{iszero } n)\ 1\ (\text{multi } n\ \text{fact}\ (\text{pred } n))$

fact = fact' fact'

- also : by fixed-point generator (almost same as self-application)