How to run Programs in System F

$\boxed{\text{CBN}}$

$$\frac{}{\lambda x{:}\tau.e\ \text{val}} \qquad \frac{}{x\ \text{val}} \qquad \frac{}{\Lambda\alpha.e\ \text{val}}$$

$$\frac{}{(\lambda x{:}\tau.e)e' \longmapsto e[e'/x]}$$

$$(\Lambda\alpha.e)\tau \longmapsto e[\tau/\alpha]$$

Example

$id : \forall\alpha.\alpha\to\alpha$ $\qquad (\alpha\to\alpha)[\text{Num}/\alpha] = \text{Num}\to\text{Num}$

$id = \Lambda\alpha.\lambda(x{:}\alpha)x$

But when using id we have to specify the type

$\underbrace{id\ \text{Num}}_{\text{Num}\to\text{Num}}$

Lemma (Progress) if $\Theta, \bullet \vdash e : \tau$ then either e val or $e \longmapsto e'$ for some e'.

Lemma (Preservation) If $\Theta, \Gamma \vdash e : \tau$ is derivable and $e \longmapsto e'$ Then $\Theta, \Gamma \vdash e' : \tau$

Theorem (Type Safety)

If $\Theta, \Gamma \vdash e : \tau$ is derivable and $e \longmapsto^* e'$, then e' is not stuck.

Prog/Pres are good way to prove type safety, but they are not necessary for type safety.

Lemma (Type substitution)

If $\Theta, \alpha ; \Gamma \vdash e : \tau$ and $\Theta \vdash \tau' : \star$ are derivable, then so is

$\Theta ; \Gamma[\tau'/\alpha] \vdash e[\tau/\alpha] : \tau[\tau'/\alpha]$

Paul Downen (Lect 5) [2018/07/06]

Evaluation Contexts

$E = \Box \mid E\,e \mid E\,\tau$

$$\frac{e \longmapsto e'}{e\,e_1 \longmapsto e'\,e_1} \qquad \frac{e \longmapsto e'}{e\,\tau \longmapsto e'\,\tau}$$

$\boxed{\text{CBV}}$

$$\frac{e'\ \mathrm{val}}{(\lambda x{:}\tau.e)\,e' \longmapsto e[e'/x]} \qquad \frac{e_1 \longmapsto e_1'}{e_1 e_2 \longmapsto e_1' e_2}$$

$$\frac{}{(\Lambda \alpha.e)\,\tau \longmapsto e[\tau/\alpha]} \qquad \frac{e_1\ \mathrm{val} \quad e_2 \longmapsto e_2'}{e_1 e_2 \longmapsto e_1 e_2'}$$

$$\frac{e \longmapsto e'}{e\,\tau \longmapsto e'\,\tau}$$

Type Erasure

Define 'erase' by induction

$\mathrm{erase}(x) = x$

$\mathrm{erase}(\lambda(x{:}\tau).e) = \lambda x.\,\mathrm{erase}(e)$

$\mathrm{erase}(e_1 e_2) = \mathrm{erase}(e_1)\,\mathrm{erase}(e_2)$

$\mathrm{erase}(\Lambda \alpha.e) = \mathrm{erase}(e)$

$\mathrm{erase}(e\,\tau) = \mathrm{erase}(e)$

Idea: erasure doesn't change the result of running a program

Want ↓

Theorem If $e \longmapsto^* v$ Then

$\mathrm{erase}(e) \longmapsto^* \mathrm{erase}(v)$

Paul Downen (Lect 5) [2018/07/06]

Want:

Theorem $e \longmapsto^* v$ implies $\mathrm{erase}(e) \longmapsto^* \mathrm{erase}(v)$

$\boxed{\text{CBN}}$ Problem: Λ is a value, but if we erase types then we erase Λ and we might no longer have a val.

Fix: $\textcircled{H}\ \bullet \vdash e : \tau$ and $\tau \neq \forall\alpha.\tau'$

Reiterate Problem: $\mathrm{erase}(\Lambda\alpha.(\lambda x.x)(\lambda x.x)) = (\lambda x.x)(\lambda x.x)$

$\boxed{\text{CBV}}$ $\Omega \longmapsto \Omega$

$\mathrm{erase}(\lambda x{:}\alpha \ldots)(\Lambda\beta.\Omega) = (\lambda x.\ldots)\ \Omega$

RHS loops forever even if ... is just a const, like 9.

Moral: occurrences of Λ matter for CBV.

Does that mean we can't have type erasure in a lang like OCaml?

Instead we could do

$\mathrm{erase}(\Lambda\alpha.e) = \lambda x.\mathrm{erase}(e)$

$\mathrm{erase}(e\ \tau) = \mathrm{erase}(e)\langle\rangle$

But is this really what we mean by type erasure?