

~minsheet~

Announcements:

- last Sam lecture.
- I'm available as a project supervisor

Functions

Syntax:

types $\tau ::= \dots$
 $\tau_1 \rightarrow \tau_2$

pre-terms $e ::= \dots$
 $\lambda x:\tau. e$
 $e_1(e_2)$

Statics:

LAM $\frac{\Gamma, x:\sigma \vdash e:\tau}{\Gamma \vdash \lambda x:\sigma. e : \sigma \rightarrow \tau}$

APP $\frac{\Gamma \vdash e_1:\sigma' \rightarrow \tau \quad \Gamma \vdash e_2:\sigma}{\Gamma \vdash e_1(e_2):\tau}$

$\lambda x:\text{Num}. \text{times}(2; x)$

$\lambda x:\text{Num}. \text{plus}(x; x)$

LAM-VAL $\frac{}{\lambda x:\sigma. e \text{ val}}$

(NUM-VAL $\frac{n \in \mathbb{N}}{\text{num}(n) \text{ val}}$)

D-APP-1 $\frac{e_1 \mapsto e'_1}{e_1(e_2) \mapsto e'_1(e_2)}$

(D-PLUS1 $\frac{e_1 \mapsto e'_1}{\text{plus}(e_1; e_2) \mapsto \text{plus}(e'_1; e_2)}$)

D-BETA $\frac{}{(\lambda x:\tau. e_1)(e_2) \mapsto e_1[e_2/x]}$

$(e_1(e_2))[e/x] \stackrel{\text{def}}{=} (e_1[e/x])(e_2[e/x])$

$(\lambda y:\tau. u)[e/x] \stackrel{\text{def}}{=} (\lambda y:\tau. u[e/x])$

Constants + functions fragment of the Simply-Typed λ -calc.

This specification of functions supports higher-order (HO) functions.

HO functions = functions that take other functions as an argument.

e.g.

$\triangleright \text{twice} :: (a \rightarrow a) \rightarrow a \rightarrow a$

$\triangleright \text{twice } f \ x = f(f \ x)$

Call-by-name Vs. Call-by-value

Call-by-name (CBN)

= where variables represent terms

Example

$$f(x) = x + x$$

$f(1+1)$ ^{1 + "hello"}
→ $(1+1) + (1+1)$
→ $2 + (1+1)$!
→ $2 + 2$
→ 4 4 steps

Call-by-value

= where variables represent values

Example:

$f(1+1)$ [!]
→ $f(2)$
→ $2 + 2$
→ 4 3 steps

Evaluation order does not affect pure results

The difference is only noticeable in the presence of effects

Example effects: printing, IO, failure, nondet, state

"side-effect" runtime

CBV language

- C
- Java
- Scala
- JS
- OCaml
- Scheme ...

Haswell = call-by-need

Optimised variant of CBN

Call-by-value λ -calc

REFERENCE: SIMPLY-TYPED λ -CALCULUS

Samantha Frohlich

No changes to statics needed. Evaluation is specified by the dynamics

$\frac{}{\Gamma, x : \sigma \vdash x : \sigma}$	$\frac{n \in \mathbb{N}}{\Gamma \vdash \text{num}[n] : \text{Num}}$	$\frac{\Gamma \vdash e_1 : \text{Num} \quad \Gamma \vdash e_2 : \text{Num}}{\Gamma \vdash \text{plus}(e_1; e_2) : \text{Num}}$
$\frac{\Gamma \vdash e_1 : \text{Num} \quad \Gamma \vdash e_2 : \text{Num}}{\Gamma \vdash \text{times}(e_1; e_2) : \text{Num}}$	$\frac{\Gamma \vdash e_1 : \sigma_1 \quad \Gamma, x : \sigma_1 \vdash e_2 : \sigma_2}{\Gamma \vdash \text{let}(e_1; x. e_2) : \sigma_2}$	
$\frac{\Gamma, x : \sigma \vdash e : \tau}{\Gamma \vdash \lambda x : \sigma. e : \sigma \rightarrow \tau}$	$\frac{\Gamma \vdash e_1 : \sigma \rightarrow \tau \quad \Gamma \vdash e_2 : \sigma}{\Gamma \vdash e_1(e_2) : \tau}$	

Figure 2: Dynamics of the constants-and-functions fragment of the simply-typed λ -calculus

$\frac{}{\lambda x : \tau. e \text{ val}}$	$\frac{n \in \mathbb{N}}{\text{num}[n] \text{ val}}$	$\frac{n_1 + n_2 = n}{\text{plus}(\text{num}[n_1]; \text{num}[n_2]) \mapsto \text{num}[n]}$
$\frac{e_1 \mapsto e'_1}{\text{plus}(e_1; e_2) \mapsto \text{plus}(e'_1; e_2)}$	$\frac{e_1 \text{ val} \quad e_2 \mapsto e'_2}{\text{plus}(e_1; e_2) \mapsto \text{plus}(e_1; e'_2)}$	
$\frac{}{\text{let}(e_1; x. e_2) \mapsto e_2[e_1/x]}$	$\frac{e_1 \mapsto e'_1}{e_1(e_2) \mapsto e'_1(e_2)}$	$\frac{}{(\lambda x : \tau. e_1)(e_2) \mapsto e_1[e_2/x]}$

$$\text{D-APP-2} \quad \frac{e_1 \text{ val} \quad e_2 \mapsto e'_2}{e_1(e_2) \mapsto e_1(e'_2)}$$

$$\text{D-LET-1} \quad \frac{e_1 \mapsto e'_1}{\text{let}(e_1; x. e_2) \mapsto \text{let}(e'_1; x. e_2)}$$

Call-by-value λ -calc-ent

Progress + Preservation hold for
CBV-STLC.

Questions

Sample reuseage of functions:

```
let f = λx:Num. plus(x x)
in      f (f (num [2]))
```