1999 CST Part IB Semantics of Prog. Langs., paper 6, q 9 AMP

Call-by-value rule:

$$\frac{\begin{cases} \langle M_1, s\rangle \Downarrow \langle fn\, x.\, M_1', s'\rangle \\ \langle M_2, s'\rangle \Downarrow \langle V_2, s''\rangle \\ \langle M_1'[V_2/x], s''\rangle \Downarrow \langle V, s'''\rangle \end{cases}}{\langle M_1 M_2, s\rangle \Downarrow \langle V, s'''\rangle}(\Downarrow_{cbv})$$

3

Call-by-name rule:

$$\frac{\begin{cases} \langle M_1, s\rangle \Downarrow \langle fn\, x.\, M_1', s'\rangle \\ \langle M_1'[M_2/x], s'\rangle \Downarrow \langle V, s''\rangle \end{cases}}{\langle M_1 M_2, s\rangle \Downarrow \langle V, s''\rangle}(\Downarrow_{cbn})$$

2

5

Write $\Downarrow_v$ (resp. $\Downarrow_n$) for the evaluation relation defined using call-by-value (resp. call-by-name) rule for application.

2 Then: $\Downarrow_v$ is neither contained in, nor contains, $\Downarrow_n$.

For example, consider

$C_0 \stackrel{def}{=}$ while true do skip

$C_1 \stackrel{def}{=} (fn\, x.\, skip)\, C_0$

$C_2 \stackrel{def}{=} (fn\, x.\, \text{if } !l = 0 \text{ then skip else } C_0)\,(l := 0)$

Then since evaluation of C_0 diverges, ie for all s $\nexists V, s'$ such that $\langle C_0, s\rangle \Downarrow_v \langle V, s'\rangle$ or $\langle C_0, s\rangle \Downarrow_n \langle V, s'\rangle$

it follows that $\boxed{\langle C_1, s\rangle \not\Downarrow_v}$

(for if $\langle C_1, s\rangle \Downarrow_v \langle V, s'''\rangle$ were provable, it would have to have been deduced using $(\Downarrow_{cbv})$, so $\langle C_0, s\rangle \Downarrow \langle V_0, s'\rangle$ would hold for some V_0, s' #).

However, $(\Downarrow_{cbv})$ implies that

$\boxed{\langle C_1, s\rangle \Downarrow_n \langle skip, s\rangle}$ (any s)

So $\Downarrow_n \not\subseteq \Downarrow_v$.

Similarly, $\boxed{\langle C_2, \{l \mapsto 1\}\rangle \not\Downarrow_n}$

(since $(C_0, \{l \mapsto 1\}) \not\Downarrow_n$)

Whereas $(\Downarrow_{cbv})$ implies

$\boxed{\langle C_2, \{l \mapsto 1\}\rangle \Downarrow_v \langle skip, \{l \mapsto 0\}\rangle}$

so $\Downarrow_v \not\subseteq \Downarrow_n$.

LFP types:

$\tau ::=$	int	integers
$\mid$	bool	booleans
$\mid$	loc	location names
$\mid$	cmd	commands
$\mid$	$\tau \to \tau$	functions

Can assign types to terms using an inductively defined relation of the form

$$\Gamma \vdash M : \tau$$

where Γ (type environment) is a finite function from variables to types. The axioms & rules generating this relation follow the structure of M. Eg rule for abstraction is

$$\frac{\Gamma, x:\tau \vdash M:\tau'}{\Gamma \vdash \mathrm{fn}x.M : \tau \to \tau'} \quad x \notin \mathrm{dom}(\Gamma)$$

for application is

$$\frac{\Gamma \vdash M_1 : \tau \to \tau' \quad \Gamma \vdash M_2 : \tau}{\Gamma \vdash M_1 M_2 : \tau'}$$

for sequencing is

$$\frac{\Gamma \vdash M_1 : \mathrm{cmd} \quad \Gamma \vdash M_2 : \mathrm{cmd}}{\Gamma \vdash M_1 ; M_2 : \mathrm{cmd}}$$

etc., etc.

Write $\boxed{M:\tau}$ to mean $\emptyset \vdash M:\tau$ is deducible (so in particular M has no free variables).

Facts • given M & τ can decide whether or not $M:\tau$ holds.

• if $M:\tau$ and $\tau \neq \mathrm{cmd}$, then forall s, s', V, $\langle M, s\rangle \Downarrow_n \langle V, s'\rangle \Rightarrow s = s'$

Thus can use type system to decide if M has type $\neq \mathrm{cmd}$, and in this case $\Downarrow_n$ evaluation is free of side-effects.

5

For $\Downarrow_v$ this property fails, because sequential composition can be defined at all types using

$$M_1 \text{ and then } M_2 \stackrel{\text{def}}{=} (\text{fn } x.\, M_2)\, M_1$$

(where $x \notin \text{freevars}(M_2)$)

For example

$$(\ell := !\ell + 1) \text{ and then } 1$$

is of type int, and its evaluation under c-b-value has a side-effect

$$\langle (\ell := !\ell + 1) \text{ and then } 1,\ \{\ell \mapsto 0\} \rangle \Downarrow_v \langle 1,\ \{\ell \mapsto 1\} \rangle.$$

2 9

(20)