Example (III): Partial correctness

Let $\mathcal{F} : State \rightharpoonup State$ be the denotation of

$$\textbf{while}\ X > 0\ \textbf{do}\ (Y := X * Y; X := X - 1)\ .$$

For all $x, y \geq 0$,

$$\mathcal{F}[X \mapsto x, Y \mapsto y] \downarrow$$
$$\implies \mathcal{F}[X \mapsto x, Y \mapsto y] = [X \mapsto 0, Y \mapsto x! \cdot y].$$

Recall that

$$\mathcal{F} = \mathit{fix}(f)$$

where $f : (State \rightharpoonup State) \to (State \rightharpoonup State)$ is given by

$$f(w) = \lambda(x,y) \in State. \begin{cases} (x,y) & \text{if } x \leq 0 \\ w(x-1, x \cdot y) & \text{if } x > 0 \end{cases}$$

Proof by Scott induction.

We consider the admissible subset of $(State \rightharpoonup State)$ given by

$$S = \left\{ w \;\middle|\; \begin{array}{l} \forall x, y \geq 0. \\ \quad w[X \mapsto x, Y \mapsto y]\downarrow \\ \qquad \Rightarrow w[X \mapsto x, Y \mapsto y] = [X \mapsto 0, Y \mapsto x! \cdot y] \end{array} \right\}$$

and show that

$$w \in S \implies f(w) \in S \ .$$

S is admissible:

(1) $\emptyset \in S$

(2) $w_0 \subseteq w_1 \subseteq \dots \subseteq w_n \subseteq \dots$ in S Then $\bigsqcup_n w_n \in S$
$(n \in \mathbb{N})$

$$S = \{ w \mid \forall x, y \geq 0.\ w[X \mapsto x, Y \mapsto y] \downarrow \Rightarrow w[X \mapsto x, Y \mapsto y] = [X \mapsto 0, Y \mapsto x! \cdot y] \}$$

w_i a chain in S.

$$\bigcup_i w_i \in S \iff \left[\forall x, y \geq 0.\ (\bigcup_i w_i)[X \mapsto x, Y \mapsto y] \downarrow \Rightarrow (\bigcup_i w_i)[X \mapsto x, Y \mapsto y] = [X \mapsto 0, Y \mapsto x! \cdot y] \right]$$

Topic 5

PCF

PCF syntax

Types

$$\tau ::= nat \mid bool \mid \tau \to \tau$$

Expressions

$$\begin{aligned} M \quad ::= \quad & \mathbf{0} \mid \mathbf{succ}(M) \mid \mathbf{pred}(M) \\ \mid \quad & \mathbf{true} \mid \mathbf{false} \mid \mathbf{zero}(M) \\ \mid \quad & x \mid \mathbf{if}\ M\ \mathbf{then}\ M\ \mathbf{else}\ M \\ \mid \quad & \mathbf{fn}\, x : \tau\, .\, M \mid M\ M \mid \mathbf{fix}(M) \end{aligned}$$

where $x \in \mathbb{V}$, an infinite set of variables.

Technicality: We identify expressions up to α-conversion of bound variables (created by the $\mathbf{fn}$ expression-former): by definition a PCF term is an α-equivalence class of expressions.

PCF typing relation, $\Gamma \vdash M : \tau$

- Γ is a type environment, *i.e.* a finite partial function mapping variables to types (whose domain of definition is denoted $dom(\Gamma)$)
- M is a term
- τ is a type.

$$\Gamma = [x_1 \mapsto \tau_1, \ldots, x_n \mapsto \tau_n]$$

$$= (x_1 : \tau_1, \ldots, x_n : \tau_n).$$

PCF typing relation, $\Gamma \vdash M : \tau$

- Γ is a type environment, *i.e.* a finite partial function mapping variables to types (whose domain of definition is denoted $dom(\Gamma)$)
- M is a term
- τ is a type.

Notation:

$M : \tau$ means M is closed and $\emptyset \vdash M : \tau$ holds.

$\mathrm{PCF}_\tau \stackrel{\text{def}}{=} \{M \mid M : \tau\}$.

PCF typing relation (sample rules)

$$(:_{\mathrm{fn}}) \quad \frac{\Gamma[x \mapsto \tau] \vdash M : \tau'}{\Gamma \vdash \mathbf{fn}\, x : \tau \,.\, M : \tau \to \tau'} \quad \text{if } x \notin dom(\Gamma)$$

$$\Gamma = [x_1 \mapsto \tau_1, \ldots, x_n \mapsto \tau_n]$$

$$\Gamma[x \mapsto \tau] = [x_1 \mapsto \tau_1, \ldots, x_n \mapsto \tau_n, x \mapsto \tau]$$

PCF typing relation (sample rules)

$$(:_{\mathrm{fn}}) \quad \frac{\Gamma[x \mapsto \tau] \vdash M : \tau'}{\Gamma \vdash \mathbf{fn}\, x : \tau \,.\, M : \tau \to \tau'} \quad \text{if } x \notin dom(\Gamma)$$

$$(:_{\mathrm{app}}) \quad \frac{\Gamma \vdash M_1 : \tau \to \tau' \quad \Gamma \vdash M_2 : \tau}{\Gamma \vdash M_1\, M_2 : \tau'}$$

$$(:_{\mathrm{fix}}) \quad \frac{\Gamma \vdash M : \tau \to \tau}{\Gamma \vdash \mathbf{fix}(M) : \tau}$$

Given F: $\underline{nat} \to \underline{nat}$ and G: $\underline{nat} \to \underline{nat} \to \underline{nat} \to \underline{nat}$ encoding f and g respectively, define H: $\underline{nat} \to \underline{nat} \to \underline{nat}$ encoding h.

Partial recursive functions in PCF

- Primitive recursion.

$$\begin{cases} h(x,0) = f(x) \\ h(x,y+1) = g(x,y,h(x,y)) \end{cases}$$

Partial rec. functions. $\mathbb{N}^n \to \mathbb{N}$

$\Updownarrow$

PCF Terms $\underbrace{\underline{nat} \to \underline{nat} \to \cdots \to \underline{nat}}_{n \text{ times.}} \to \underline{nat}$

$$\begin{cases} h(x,0) = f(x) \\ h(x,y+1) = g(x,y,h(x,y)) \end{cases}$$

$$\begin{aligned} H\ x\ y = \ & \underline{\text{if}}\ \underline{\text{zero}}(y) \\ & \underline{\text{then}}\ F\ x \\ & \underline{\text{else}}\ G\ x\ (\underline{\text{pred}}\ y)\ (H\ x\ (\underline{\text{pred}}\ y)) \end{aligned}$$

Consider $M = \text{fn}\,H.\ \text{fn}\,x.\ \text{fn}\,y.$

$$\underline{\text{if}}\ \underline{\text{zero}}(y)\ \underline{\text{then}}\ Fx$$
$$\text{else}\ G\ x\ (\underline{\text{pred}}\ y)\ (H\ x\ (\text{pred}\ y))$$

$$H =^{\text{def}} \underline{\text{fix}}(M)$$

Partial recursive functions in PCF

- Primitive recursion.

$$\begin{cases} h(x, 0) = f(x) \\ h(x, y+1) = g(x, y, h(x, y)) \end{cases}$$

- Minimisation.

$$m(x) \;=\; \text{the least } y \geq 0 \text{ such that } k(x, y) = 0$$

Consider an encoding of k, say $K : \underline{nat} \to \underline{nat} \to \underline{nat}$

Define a PCF term testing for k

$$T\ x\ y = \underline{\text{if}}\ \underline{\text{zero}}\,(K\ x\ y)\ \underline{\text{then}}\ y\ \underline{\text{else}}\ T\ x\ (\underline{\text{succ}}\ y)$$

$$K = \text{fn}\,x.\ \underline{\text{fix}}\,(\text{fn}\,T.\ \text{fn}\,x.\ \text{fn}\,y.\ \underline{\text{if}}\ \underline{\text{zero}}\,(K\ x\ y)\ \underline{\text{then}}\ y\ \underline{\text{else}}\ T\,(x\,(\underline{\text{succ}}\,y)))\,x\,0$$

PCF evaluation relation

takes the form

$$M \Downarrow_\tau V$$

where

- τ is a PCF type
- $M, V \in \mathrm{PCF}_\tau$ are closed PCF terms of type τ
- V is a value,

$$V ::= \mathbf{0} \mid \mathbf{succ}(V) \mid \mathbf{true} \mid \mathbf{false} \mid \mathbf{fn}\, x : \tau \,.\, M.$$

PCF evaluation (sample rules)

$$(\Downarrow_{\mathrm{val}}) \quad V \Downarrow_\tau V \qquad (V \text{ a value of type } \tau)$$

$$(\Downarrow_{\mathrm{cbn}}) \quad \frac{M_1 \Downarrow_{\tau \to \tau'} \mathbf{fn}\, x : \tau \,.\, M_1' \qquad M_1'[M_2/x] \Downarrow_{\tau'} V}{M_1\, M_2 \Downarrow_{\tau'} V}$$

call-by-name!

PCF evaluation (sample rules)

$(\Downarrow_{\mathrm{val}})\quad V \Downarrow_\tau V$ $\qquad$ (V a value of type τ)

$$(\Downarrow_{\mathrm{cbn}})\quad \frac{M_1 \Downarrow_{\tau\to\tau'} \mathbf{fn}\, x : \tau \,.\, M_1' \qquad M_1'[M_2/x] \Downarrow_{\tau'} V}{M_1\, M_2 \Downarrow_{\tau'} V}$$

$$(\Downarrow_{\mathrm{fix}})\quad \frac{M\,(\mathbf{fix}(M)) \Downarrow_\tau V}{\mathbf{fix}(M) \Downarrow_\tau V}$$

$$\Omega_\tau =^{def} \underline{\mathbf{fix}}(\underline{\mathbf{fn}}\, x : \tau .\, x) \in \underline{PCF}_\tau$$

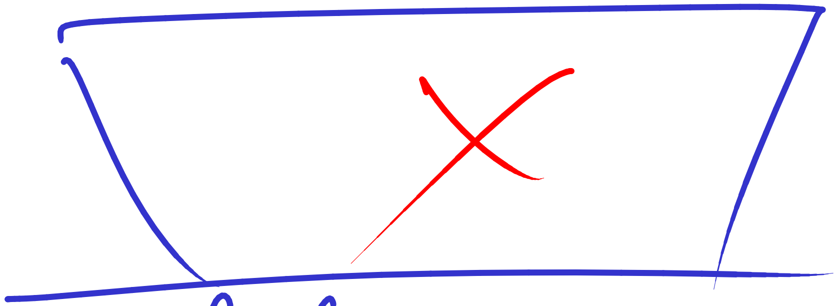
<u>NB</u>: There is no value V for which

$$\Omega_\tau \Downarrow V$$

Suppose $\Omega_\tau \Downarrow V$ for some V.
Then we have a derivation

$$\Omega_\tau \Downarrow V$$

Consider one such of <u>minimal size</u>



$$\mathrm{fix}(\mathrm{fn}\,x.x)$$

$$\dfrac{\dfrac{\mathrm{fn}\,x.x \Downarrow \mathrm{fn}\,x.x \qquad x[\mathrm{fix}(\mathrm{fn}\,x.x)/x] \Downarrow v}{(\mathrm{fn}\,x.x)(\mathrm{fix}(\mathrm{fn}\,x.x)) \Downarrow v}}{\mathrm{fix}(\mathrm{fn}\,x.x) \Downarrow v}$$

NB For pred we have that there is no V such that

$$\text{pred}(0) \Downarrow V$$

$$\frac{M \Downarrow \text{succ}(V)}{\text{pred}(M) \Downarrow V}$$

NB: pred(0) and Ω_{nat} have the same operational behaviour

Contextual equivalence

Two phrases of a programming language are contextually equivalent if any occurrences of the first phrase in a complete program can be replaced by the second phrase without affecting the observable results of executing the program.

Contextual equivalence of PCF terms

Given PCF terms M_1, M_2, PCF type τ, and a type environment Γ, the relation $\boxed{\Gamma \vdash M_1 \cong_{\mathrm{ctx}} M_2 : \tau}$ is defined to hold iff

- Both the typings $\Gamma \vdash M_1 : \tau$ and $\Gamma \vdash M_2 : \tau$ hold.
- For all PCF contexts $\mathcal{C}$ for which $\mathcal{C}[M_1]$ and $\mathcal{C}[M_2]$ are closed terms of type γ, *where* $\gamma = nat$ *or* $\gamma = bool$, and for all values $V : \gamma$,

$$\mathcal{C}[M_1] \Downarrow_\gamma V \;\Leftrightarrow\; \mathcal{C}[M_2] \Downarrow_\gamma V.$$