PCF syntax

Types

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

Expressions

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

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 (sample rules)

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

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

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

<u>Idea</u> $F = \underline{fix}(\lambda f. \lambda x. \sim\sim f \sim\sim x \sim)$

The recursive function $F x = \cdots\cdots F \cdots x \cdots$

Suppose $F : nat \to nat$ $\quad G : nat \to nat \to nat \to nat$

$\boxed{?}$ $H : nat \to nat \to nat$?

Partial recursive functions in PCF

- Primitive recursion.

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

Idea

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

Def:

$H = \underline{\text{fix}}\ (\lambda h.\lambda x.\lambda y.\ \underline{\text{if}}\ (\text{zero}\ y)\ \underline{\text{then}}\ F(x)$
$\quad \underline{\text{else}}\ G\ x\ (\underline{\text{pred}}\ y)\ (h\ x\ (\underline{\text{pred}}\ y)))$

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.

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

Suppose $K : nat \to nat \to nat$.

Define $M : nat \to nat$?

idea

$M\,x = F\,x\,0$

$F\,x\,y =$ if zero$(K\,x\,y)$ then y

else $F\,x\,(\text{succ}\,y)$

def

$F = \text{fix}(\lambda f.\lambda x.\lambda y.$ if (zero $K\,x\,y$) then y

else $f\,x\,(\text{succ}\,y))$

$M =$ fn $x{:}\text{nat}.\ F\,x\,0$

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,

$$V ::= \underbrace{\mathbf{0} \mid \mathbf{succ}(V) \mid \mathbf{true} \mid \mathbf{false}}_{\text{values of ground types = nat \& bool.}} \mid \overbrace{\mathbf{fn}\, x : \tau \,.\, M}^{\text{closure}}.$$

PCF evaluation (sample rules)

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

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

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

$M_1 \cong_{ctx} M_2 \iff \forall\, \mathcal{C}[\;] : \mathcal{C}[M_1]$ and $\mathcal{C}[M_2]$ are indistinguishable

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.

Remark $(nat \to nat) \to (nat \to nat)$

$fn\ F : nat \to nat.\ F$

$fn\ F : nat \to nat.\ fn\ x : nat.\ F(x)$

two values for the identity function.

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 $V : \gamma$,

ground type

!

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

PCF denotational semantics — aims

- PCF types $\tau \;\mapsto\;$ domains $[\![\tau]\!]$.

$$[\![bool]\!] = \left(\begin{tikzcd}[column sep=small] true \arrow[dr, dash] & & false \arrow[dl, dash] \\ & \bot & \end{tikzcd}\right)$$

$$[\![nat]\!] = \left(\begin{tikzcd}[column sep=small] 0 \arrow[drr, dash] & 1 \arrow[dr, dash] & 2 \arrow[d, dash] & \cdots & n \arrow[dll, dash] & \cdots \\ & & \bot & & & \end{tikzcd}\right) \quad \text{new}$$

$$[\![\tau_1 \to \tau_2]\!] = \left([\![\tau_1]\!] \to [\![\tau_2]\!]\right)$$

PCF denotational semantics — aims

- PCF types $\tau \mapsto$ domains $[\![\tau]\!]$.

a term in the empty context.

- Closed PCF terms $M : \tau \mapsto$ elements $[\![M]\!] \in [\![\tau]\!]$.

  Denotations of open terms will be continuous functions.

$$\Gamma \vdash M : \tau \quad \leadsto \quad [\![\Gamma]\!] \xrightarrow{[\![M]\!]} [\![\tau]\!]$$

a domain of environments

The domain associated to a context

$$\Gamma \equiv (x_1 : \tau_1, x_2 : \tau_2, \dots ; x_n : \tau_n)$$

$\llbracket \Gamma \rrbracket$ a domain of environments

$$\rho \in \left(\llbracket \tau_1 \rrbracket \times \llbracket \tau_2 \rrbracket \times \cdots \times \llbracket \tau_n \rrbracket \right)$$

$$\rho = (d_1, d_2, \dots, d_n) \quad d_i \in \llbracket \tau_i \rrbracket$$

An equivalent definition:

environments are functions that to each x_i assign a $d_i \in \llbracket \tau_i \rrbracket$.

PCF denotational semantics — aims

- PCF types $\tau \;\mapsto\;$ domains $[\![\tau]\!]$.
- Closed PCF terms $M : \tau \;\mapsto\;$ elements $[\![M]\!] \in [\![\tau]\!]$.

  Denotations of open terms will be continuous functions.
- Compositionality.

  $\forall \mathcal{C}$

  In particular: $[\![M]\!] = [\![M']\!] \;\Rightarrow\; [\![\mathcal{C}[M]]\!] = [\![\mathcal{C}[M']]\!]$.

  $\begin{array}{ccc} \| & & \| \\ \cdots [\![M]\!] \cdots & & \cdots [\![M']\!] \cdots \end{array}$

  the same interpretation!

PCF denotational semantics — aims

- PCF types $\tau \;\mapsto\;$ domains $[\![\tau]\!]$.

- Closed PCF terms $M : \tau \;\mapsto\;$ elements $[\![M]\!] \in [\![\tau]\!]$.

  Denotations of open terms will be continuous functions.

- Compositionality.

  In particular: $[\![M]\!] = [\![M']\!] \;\Rightarrow\; [\![\mathcal{C}[M]]\!] = [\![\mathcal{C}[M']]\!]$.

- Soundness.

  For any type τ, $M \Downarrow_\tau V \;\Rightarrow\; [\![M]\!] = [\![V]\!]$.

- Adequacy.

  For $\tau = bool$ or nat, $[\![M]\!] = [\![V]\!] \in [\![\tau]\!] \;\Longrightarrow\; M \Downarrow_\tau V$.

Theorem. *For all types* τ *and closed terms* $M_1, M_2 \in \mathrm{PCF}_\tau$, *if* $[\![M_1]\!]$ *and* $[\![M_2]\!]$ *are equal elements of the domain* $[\![\tau]\!]$, *then* $M_1 \cong_{\mathrm{ctx}} M_2 : \tau$.

Proof principle:

$$\frac{[\![M_1]\!] = [\![M_2]\!]}{M_1 \cong_{\mathrm{ctx}} M_2}$$

Theorem. *For all types* τ *and closed terms* $M_1, M_2 \in \mathrm{PCF}_\tau$, *if* $[\![M_1]\!]$ *and* $[\![M_2]\!]$ *are equal elements of the domain* $[\![\tau]\!]$, *then* $M_1 \cong_{\mathrm{ctx}} M_2 : \tau$.

Proof. $\forall \mathcal{C}$.

$[\![\mathcal{C}[M_1]]\!] = [\![\mathcal{C}[M_2]]\!]$

$$\mathcal{C}[M_1] \Downarrow_{nat} V \Rightarrow [\![\mathcal{C}[M_1]]\!] = [\![V]\!] \quad \text{(soundness)}$$

$$\Rightarrow [\![\mathcal{C}[M_2]]\!] = [\![V]\!] \quad \text{(compositionality on } [\![M_1]\!] = [\![M_2]\!]\text{)}$$

$$\Rightarrow \mathcal{C}[M_2] \Downarrow_{nat} V \quad \text{(adequacy)}$$

and symmetrically. $\square$

Proof principle

To prove

$$M_1 \cong_{\mathrm{ctx}} M_2 : \tau$$

it suffices to establish

$$[\![M_1]\!] = [\![M_2]\!] \text{ in } [\![\tau]\!]$$

$\boxed{?}$ The proof principle is sound, but is it complete? That is, is equality in the denotational model also a necessary condition for contextual equivalence?