$\perp \in S$

$\forall\, d_0 \sqsubseteq d_1 \sqsubseteq \ldots \sqsubseteq d_n \sqsubseteq \ldots$ in S. $\Rightarrow \bigsqcup_n d_n \in S$

Scott's Fixed Point Induction Principle

Let $f : D \to D$ be a continuous function on a domain D.

For any admissible subset $S \subseteq D$, to prove that the least fixed point of f is in S, *i.e.* that

$$\mathit{fix}(f) \in S \ ,$$

it suffices to prove

$$\forall d \in D \; (d \in S \;\Rightarrow\; f(d) \in S) \ .$$

$$\frac{d \in S \Rightarrow f(d) \in S}{\mathit{fix}(f) \in S}$$

Example (I): Least pre-fixed point property

Let D be a domain and let $f : D \to D$ be a continuous function.

$$\forall d \in D.\ f(d) \sqsubseteq d \implies fix(f) \sqsubseteq d$$

$x \sqsubseteq d \Rightarrow fx \sqsubseteq fd \sqsubseteq d \Rightarrow fx \sqsubseteq d$.

$$\frac{\dfrac{x \sqsubseteq d \Rightarrow f(x) \sqsubseteq d}{x \in \downarrow(d) \Rightarrow f(x) \in \downarrow(d)}}{\dfrac{fix(f) \in \downarrow(d)}{fix(f) \sqsubseteq d}}$$

admissible

$S = \downarrow(d)$
$= \{x \mid x \sqsubseteq d\}$

Building chain-closed subsets (III)

Logical operations:

- If $S, T \subseteq D$ are chain-closed subsets of D then

$$S \cup T \qquad \text{and} \qquad S \cap T$$

  are chain-closed subsets of D.

- If $\{ S_i \}_{i \in I}$ is a family of chain-closed subsets of D indexed by a set I, then $\bigcap_{i \in I} S_i$ is a chain-closed subset of D.

- If a property $P(x, y)$ determines a chain-closed subset of $D \times E$, then the property $\forall x \in D.\ P(x, y)$ determines a chain-closed subset of E.

$\mathbb{L} \to \mathbb{Z}$ where $\mathbb{L} = \{X, Y\}$.

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 $x, y \geq 0$,

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

partial correctness.

recall $\mathcal{F} = \underline{fix}(f)$,

$w \in S \Rightarrow f(w) \in S$

$\forall f (\forall x y.\ w(x,y)\downarrow \Rightarrow w(x,y) = (0, !x \cdot y))$ ①

$\Rightarrow \forall x y.\ fw(x,y)\downarrow \Rightarrow fw(x,y)$ ②

$= (0, !x \cdot y)$

Recall that

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

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

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

value of X   value of Y

Assume ① & ②. Case $x=0$ ...

Case $x>0$: $fw(x,y) = w(x-1, x \cdot y)$

// by ①

$(0, !x \cdot y) = (0, !(x-1) \cdot (x \cdot y))$

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

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

$$\overline{\qquad\qquad\qquad\qquad\qquad\qquad}$$

$$\underline{fix}(f) \in S$$

Topic 5

PCF

PCF syntax

Types $[\![nat]\!] = \mathbb{N}_\perp$, $[\![bool]\!] = \mathbb{B}_\perp$, $[\![\tau \to \sigma]\!]$

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

$= ([\![\tau]\!] \to [\![\sigma]\!])$.

PCF syntax

Types

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

Expressions

$$M \quad ::= \quad \mathbf{0} \mid \mathbf{succ}(M) \mid \mathbf{pred}(M)$$

PCF syntax

Types

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

Expressions

ML

fn $x:\tau \Rightarrow M$

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

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

application

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.

$(x_1 : \tau_1, x_2 : \tau_2, \ldots, x_n : \tau_n)$

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.

$[\![x_1 : \tau_1, x_2 : \tau_2, \ldots, x_n : \tau_n]\!]$

$= ([\![\tau_1]\!] \times [\![\tau_2]\!] \times \cdots \times [\![\tau_n]\!])$

$[\![x_1 : \tau_1, \ldots, x_n : \tau_n \vdash M : \tau]\!]$

$[\![\tau]\!]$

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.

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

PCF typing relation (sample rules)

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

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

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

Partial recursive functions in PCF

- Primitive recursion.

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

$$\underline{\mathbf{fix}}\left(\mathbf{fn}\, h.\, \mathbf{fn}\, z_1.\, \mathbf{fn}\, z_2.\ \underline{\mathbf{if}}\ \underline{\mathbf{zero}}(z_2)\ \underline{\mathbf{then}}\ f(z_1)\ \underline{\mathbf{else}}\ g\ z_1\ (\underline{\mathbf{pred}}\ z_2)\ (h\ z_1\ (\underline{\mathbf{pred}}\ z_2))\right)$$

In PCF we can write programs of type
(nat → bool) → bool That can be thought of as taking input boolean streams.

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.

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

$m(x) = t(x,0)$
$t(x,y) =$ if $k(x,y)=0$ then y
else $t(x,y+1)$

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

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