Topic 5

PCF

PCF syntax

Types

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

E.g. $bool \to (bool \to bool)$

$(nat \to bool) \to bool$

PCF syntax

Types

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

E.g. $bool \to (bool \to bool)$

$(nat \to bool) \to bool$

$\to$ is <u>right</u> associative:

"$\tau_1 \to \tau_2 \to \tau_3$" means $\tau_1 \to (\tau_2 \to \tau_3)$

PCF syntax

Types

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

Expressions

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

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.

Application is left associative:
"$M_1\, M_2\, M_3$" means $(M_1\, M_2)\, M_3$

Whereas in OCaml one might write

```
let rec f x = if x=0 then 1 else x*f(x-1) in f 42
```

in PCF one has to write

$$(\mathbf{fix}(\mathbf{fn}\ f : nat \to nat.\ \mathbf{fn}\ x : nat.$$
$$\quad \mathbf{if}\ zero(x)\ \mathbf{then}\ succ(0)$$
$$\quad \mathbf{else}\ times\ x\ (f\ (pred\ (x)))\ ))\ succ^{42}(0)$$

Whereas in OCaml one might write

let rec f x = if x=0 then 1 else x*f(x-1) in f 42

in PCF one has to write

$$(\mathbf{fix}(\mathbf{fn}\ f : nat \to nat .\ \mathbf{fn}\ x : nat .\ \mathbf{if}\ \mathbf{zero}(x)\ \mathbf{then}\ \mathbf{succ}(\mathbf{0})\ \mathbf{else}\ times\ x\ (f(\mathbf{pred}(x))))) \ \mathbf{suc}^{42}(\mathbf{0})$$

$$\boxed{\begin{array}{l}\text{where } \mathbf{suc}^{42}(\mathbf{0}) \triangleq \underbrace{\mathbf{suc}(\mathbf{suc}(\ldots\mathbf{suc}}_{42\ \mathbf{suc}\text{'s}}(\mathbf{0})\ldots))\\ \text{\& } times \text{ is as on p47 of the notes.}\end{array}}$$

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.

E.g. fix(fn x:τ. x) = fix(fn y:τ. y)

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.

if this contains distinct variables $x_1, x_2, \ldots, x_n$ and $\Gamma(x_i) = \tau_i$, we sometimes write Γ as $\{x_1 : \tau_1, x_2 : \tau_2, \ldots, x_n : \tau_n\}$

PCF typing relation (sample rules)

See Fig. 2 page 40

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

$dom(\Gamma[x \mapsto \tau]) = dom\,\Gamma \cup \{x\}$

$\Gamma[x \mapsto \tau]$ maps x to τ and otherwise acts like Γ

PCF typing relation (sample rules)

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

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

PCF typing relation (sample rules)

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

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

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

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.

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

i.e. $fv(M) = \emptyset$

where...

$fv(M)$ – set of free variables of M is defined by:

$fv(0) = fv(\text{true}) = fv(\text{false}) = \emptyset$
$fv(\text{succ}(M)) = fv(\text{pred}(M)) = fv(\text{zero}(M)) = fv(\text{fix}(M)) = fv(M)$
$fv(\text{if } M \text{ then } M' \text{ else } M'') = fv(M) \cup fv(M') \cup fv(M'')$
$fv(M\ M') = fv(M) \cup fv(M')$
$fv(x) = \{x\}$
$fv(\text{fn } x : \tau.\ M) = \{x' \in fv(M) \mid x' \neq x\}$

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,

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

See Fig. 3 page 43

PCF evaluation (sample rules)

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

PCF evaluation (sample rules)

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

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

PCF evaluation (sample rules)

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

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

<u>substitution</u> (capture-avoiding — but since M_2 is closed there can be no capture)

<u>NB</u> if $\Gamma[x \mapsto \tau] \vdash M_1' : \tau'$
& $\Gamma \vdash M_2 : \tau$, then $\Gamma \vdash M_1'[M_2/x] : \tau'$
(see Proposition 5.3.1 (ii))

PCF evaluation (sample rules)

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

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

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

PCF evaluation

$$(\Downarrow_{pred}) \quad \frac{M \Downarrow_{nat} succ(V)}{pred(M) \Downarrow_{nat} V}$$

is the only rule for $pred$.

Since $0 \Downarrow_{nat} V$ only holds for $V = 0$

we conclude that $pred(0) \not\Downarrow_{nat} V$

(Making $pred(0)$ not evaluate to anything is a somewhat arbitrary choice.)

Defining $\boxed{\Omega_\tau \triangleq \mathrm{fix}(\mathrm{fn}\ x : \tau .\ x)}$

we get

$\Omega_\tau : \tau$ (proof – easy)

& $\nexists V .\ \Omega_\tau \Downarrow_\tau V$ (proof ...

If $\mathrm{fix}(\mathrm{fn}\, x:\tau.\, x) \Downarrow_\tau V$ had any proof, then we could find one of smallest height, n say, and it must look like

$$\dfrac{\dfrac{\dfrac{}{\mathrm{fn}\, x:\tau.\, x \Downarrow_{\tau\to\tau} \mathrm{fn}\, x:\tau.\, x}\,(\Downarrow_{val}) \qquad \begin{array}{c}\vdots \\ x[\mathrm{fix}(\mathrm{fn}\, x:\tau.\, x)/x] \Downarrow_\tau V\end{array}}{(\mathrm{fn}\, x:\tau.\, x)(\mathrm{fix}(\mathrm{fn}\, x:\tau.\, x)) \Downarrow_\tau V}\,(\Downarrow_{cbn})}{\mathrm{fix}(\mathrm{fn}\, x:\tau.\, x) \Downarrow_\tau V}\,(\Downarrow_{fix})$$

If $\mathrm{fix}(\mathrm{fn}\,x:\tau.\,x) \Downarrow_\tau V$ had any proof, then we could find one of smallest <u>height</u>, n say, and it must look like

$$\dfrac{\dfrac{\dfrac{}{\mathrm{fn}\,x:\tau.\,x \Downarrow_{\tau\to\tau} \mathrm{fn}\,x:\tau.\,x}\,(\Downarrow_{val}) \qquad \begin{array}{c}\vdots\\ \mathrm{fix}(\mathrm{fn}\,x:\tau.\,x) \Downarrow_\tau V\end{array}}{(\mathrm{fn}\,x:\tau.\,x)(\mathrm{fix}(\mathrm{fn}\,x:\tau.\,x)) \Downarrow_\tau V}\,(\Downarrow_{cbn})}{\mathrm{fix}(\mathrm{fn}\,x:\tau.\,x) \Downarrow_\tau V}\,(\Downarrow_{fix})$$

If $\mathrm{fix}(\mathrm{fn}\,x:\tau.\,x) \Downarrow_\tau V$ had any proof, then we could find one of smallest height, n say, and it must look like

this is a proof of height $< n$, contradicting this

$$
\dfrac{
  \dfrac{
    \dfrac{}{\mathrm{fn}\,x:\tau.\,x \Downarrow_{\tau\to\tau} \mathrm{fn}\,x:\tau.\,x}\,(\Downarrow_{\mathrm{val}})
    \qquad
    \boxed{\dfrac{\vdots}{\mathrm{fix}(\mathrm{fn}\,x:\tau.\,x) \Downarrow_\tau V}}
  }{(\mathrm{fn}\,x:\tau.\,x)(\mathrm{fix}(\mathrm{fn}\,x:\tau.\,x)) \Downarrow_\tau V}\,(\Downarrow_{\mathrm{cbn}})
}{\mathrm{fix}(\mathrm{fn}\,x:\tau.\,x) \Downarrow_\tau V}\,(\Downarrow_{\mathrm{fix}})
$$

So no such proof can exist.

PCF

"Programing Computable Functions"

?

We represent numbers $n \in \mathbb{N} = \{0, 1, 2, \ldots\}$ by closed values $\mathrm{suc}^n(0) : nat$ in PCF

$$\begin{cases} \mathrm{suc}^0(0) = 0 \\ \mathrm{suc}^{n+1}(0) = \mathrm{suc}(\mathrm{suc}^n(0)) \end{cases}$$

FACT For any computable partial function $f : \mathbb{N} \rightharpoonup \mathbb{N}$ there is a closed PCF term $F : nat \to nat$ such that for all $n, m \geq 0$

$$F(\mathrm{suc}^m(0)) \Downarrow_{nat} \mathrm{suc}^n(0)$$

if & only if

f is defined at m & $f(m) = n$

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 recursive functions in PCF

- Primitive recursion.

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

if f is programmed in PCF by F : nat → nat

& g " " " " " G : nat → nat → nat → nat

then h can be programmed by :

$$\boxed{\begin{array}{l} \text{fix(fn } h : \text{nat} \to \text{nat} \to \text{nat. fn } x : \text{nat. fn } y : \text{nat.} \\ \quad \text{if zero}(y) \text{ then } F\,x \text{ else } G\,x\,(\text{pred } y)(h\,x\,(\text{pred } y))) \end{array}}$$

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.

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

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.

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

If k is programmed in PCF by $K : nat \to nat \to nat$
then m can be programmed by $\boxed{fn\ x : nat.\ M'\, x\, 0}$
where $M' \triangleq fix(fn\ m' : nat \to nat \to nat.\ fn\ x : nat.\ fn\ y : nat.$
$if\ zero(K\,x\,y)\ then\ y\ else\ m'\,x\,(succ\ y))$