Topic 5

PCF

PCF syntax

Types

$$\tau ::= \underline{nat} \mid \underline{bool} \mid \underline{\tau \to \tau}$$

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

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

PCF syntax

Types

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

Expressions

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

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.

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.

<u>Convention</u>

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

The partial function with domain $\{x_1, x_2, \ldots, x_n\}$ that maps x_i to τ_i $(i = 1, \ldots, n)$

Notation:

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

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

THE TYPING RELATION

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

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)

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

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

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.

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

A term H encoding the function h will have curried type $\underline{nat} \to \underline{nat} \to \underline{nat}$.

Partial recursive functions in PCF

- Primitive recursion.

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

We informally want that H is recursively defined by

H = fn x. fn y.
if (zero y) then (F x)
else G x (pred y) (H x (pred y))

Given F: nat → nat and G: nat → nat → nat → nat

encoding f and g.

Partial recursive functions in PCF

- Primitive recursion.

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

Formally,

H = fix (fn h . fn x . fn y .

if (zero y) then (F x)

else G x (pred(y)) (h x (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.

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

Given a term $K : \underline{nat} \to \underline{nat} \to \underline{nat}$ encoding the function k, we want a term $M : \underline{nat} \to \underline{nat}$ encoding the function m.

Consider the following informal recursive definition that iteratively tests values of K being 0:

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

Then $M = \underline{fn}\,x.\ T\ x\ 0$ where formally

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

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)

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

PCF evaluation (sample rules)

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

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

PCF evaluation (sample rules)

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

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

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

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

There is no value to which <u>pred</u>(0) evaluates to.

Contextual equivalence

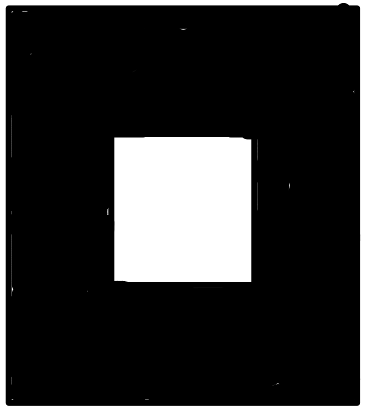
Two phrases of a programming language are contextually equivalent if any occurrences of the first phrase in a complete program can be replaced by the second phrase without affecting the observable results of executing the program.

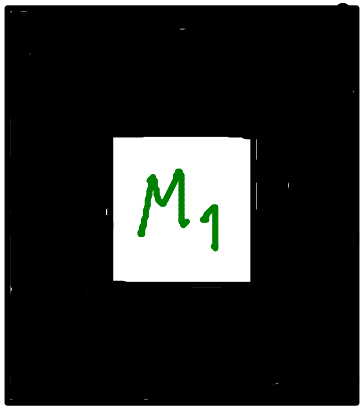
Intuitively, two program phrases are contextually equivalent whenever there is no observable computational difference between running either of them within any given complete program.

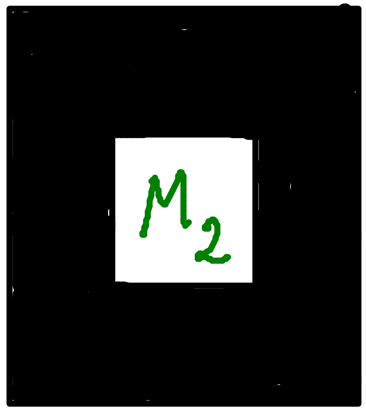
THE IDEA OF CONTEXTUAL EQUIVALENCE

$M_1 \cong_{ctx} M_2$

iff

for all program contexts 

running 

and running 

is computationally indistinguishable

NB: A context $\mathcal{C}$ is a term with a hole $[-]$.

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 $V : \gamma$,

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

NB: $\mathcal{C}[M]$ is the term obtained by filling (or instantiating) the hole of $\mathcal{C}$ with M.

Example:

There is a closed term of every type.

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

Suppose that

$$\underline{fix}(\underline{fn}\, x:\tau.\, x) \Downarrow v$$

Consider a derivation of minimal height

$$\frac{\qquad\qquad}{fix(\underline{fn}\, x:\tau.\, x) \Downarrow v}$$

$$
\dfrac{
  \dfrac{\underline{\mathrm{fn}}\ x{:}\tau.\,x \Downarrow \underline{\mathrm{fn}}\ x{:}\tau.\,x \qquad \dfrac{}{\begin{array}{c}\Omega_\tau \\ \shortparallel \\ x[\Omega_\tau/x] \Downarrow V\end{array}}}
  {(\underline{\mathrm{fn}}\ x{:}\tau.\,x)(\Omega_\tau) \Downarrow V}
}{
  \underline{\mathrm{fix}}(\mathrm{fn}\ x{:}\tau.\,x) \Downarrow V
}
$$

There is no value V such that $\Omega_\tau \Downarrow V$.

$\forall$ values V.

$$(\underline{\mathrm{pred}}(0) \Downarrow V) \iff (\Omega_{\mathrm{nat}} \Downarrow V)$$

CONJECTURE

$$\vdash \underline{\mathrm{pred}}(0) \cong_{\mathrm{ctx}} \Omega_{\mathrm{nat}} : \mathrm{nat}$$

PCF denotational semantics — aims

PCF denotational semantics — aims

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

Γ typing environment $\longmapsto$ $[\![\Gamma]\!]$ domain

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.

$$[\![\Gamma \vdash M : \tau]\!] : [\![\Gamma]\!] \longrightarrow [\![\tau]\!]$$

continuous function

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

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, $M \Downarrow_\tau V \;\Rightarrow\; [\![M]\!] = [\![V]\!]$.

PCF denotational semantics — aims

- PCF types $\tau \quad \mapsto \quad$ domains $[\![\tau]\!]$.
- Closed PCF terms $M : \tau \quad \mapsto \quad$ 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, $[\![M]\!] = [\![V]\!] \in [\![\tau]\!] \implies M \Downarrow_\tau V$.

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

Example

$$[\![\underline{\mathrm{pred}}(0)]\!] = [\![\Omega_{nat}]\!]$$

$$\Longrightarrow$$

$$\underline{\mathrm{pred}}(0) \cong_{ctx} \Omega_{nat} : nat$$

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

For $\mathcal{C}[-]$ of observable type and a value V

$$\mathcal{C}[M_1] \Downarrow V \Rightarrow [\![\mathcal{C}[M_1]]\!] = [\![V]\!]$$

$$\Rightarrow [\![\mathcal{C}[M_2]]\!] = [\![V]\!]$$

$$\Rightarrow \mathcal{C}[M_2] \Downarrow V$$

$\boxtimes$

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.

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

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

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

and symmetrically. $\square$

Proof principle

To prove

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

it suffices to establish

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

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

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?