Semantics of Programming Languages

Andrzej Murawski and Nikos Tzevelekos

Lecture 2: Equivalence in PCF, Complete Partial Orders

PCF syntax

$$\frac{((x : A) \in \Gamma)}{\Gamma \vdash x : A} \qquad \frac{}{\Gamma \vdash \mathtt{t} : \mathtt{bool}} \qquad \frac{}{\Gamma \vdash \mathtt{f} : \mathtt{bool}} \qquad \frac{}{\Gamma \vdash n : \mathtt{nat}}$$

$$\frac{}{\Gamma \vdash \mathtt{succ} : \mathtt{nat} \to \mathtt{nat}} \qquad \frac{}{\Gamma \vdash \mathtt{pred} : \mathtt{nat} \to \mathtt{nat}} \qquad \frac{}{\Gamma \vdash \mathtt{zero?} : \mathtt{nat} \to \mathtt{bool}}$$

$$\frac{(A_b \in \{\mathtt{bool}, \mathtt{nat}\})}{\Gamma \vdash \mathtt{cond}_{A_b} : \mathtt{bool} \to (A_b \to (A_b \to A_b))} \qquad \frac{}{\Gamma \vdash \mathbf{Y}_A : (A \to A) \to A}$$

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x^A.M : A \to B} \qquad \frac{\Gamma \vdash M : A \to B \qquad \Gamma \vdash N : A}{\Gamma \vdash M\, N : B}$$

Observe that terms are the **typable** elements of the following grammar.

$$M, N \quad ::= \quad x \;\mid\; \mathsf{c} \;\mid\; \lambda x^A.M \;\mid\; M\, N$$

PCF operational semantics

Values: $V \quad ::= \quad \mathsf{c} \mid \lambda x.M \qquad (\mathsf{FV}(\lambda x.M) = \emptyset)$

$$\frac{}{V \Downarrow V} \qquad \frac{M \Downarrow V' \quad V'N \Downarrow V}{M\,N \Downarrow V}\ M \text{ not a value} \qquad \frac{M[N/x] \Downarrow V}{(\lambda x.M)N \Downarrow V}$$

$$\frac{M \Downarrow n}{\mathtt{succ}\,M \Downarrow n+1} \qquad \frac{M \Downarrow n+1}{\mathtt{pred}\,M \Downarrow n} \qquad \frac{M \Downarrow 0}{\mathtt{pred}\,M \Downarrow 0}$$

$$\frac{M \Downarrow n+1}{\mathtt{zero?}\ M \Downarrow \mathtt{f}} \qquad \frac{M \Downarrow 0}{\mathtt{zero?}\ M \Downarrow \mathtt{t}} \qquad \frac{M(\mathbf{Y}_A M) \Downarrow V}{\mathbf{Y}_A M \Downarrow V}$$

$$\frac{M \Downarrow \mathtt{t}}{\mathtt{cond}_{A_b}\,M \Downarrow \lambda x^{A_b}.\lambda y^{A_b}.x} \qquad \frac{M \Downarrow \mathtt{f}}{\mathtt{cond}_{A_b}\,M \Downarrow \lambda x^{A_b}.\lambda y^{A_b}.y}$$

Partiality

Because of divergence, in PCF we can also represent **partial functions**.

We say that a partial function $\phi : \mathbb{N}^m \rightharpoonup \mathbb{N}$ is **definable** by a term $\vdash M : \underbrace{\mathtt{nat} \to \cdots \to \mathtt{nat}}_{m} \to \mathtt{nat}$ if, for every tuple $\langle n_1, \ldots, n_m \rangle \in \mathbb{N}^m$:

$$M\, n_1 \ldots n_m \begin{cases} \Downarrow \phi(n_1, \ldots, n_m) & \text{if } \phi(n_1, \ldots, n_m) \text{ defined} \\ \Uparrow & \text{if } \phi(n_1, \ldots, n_m) \text{ undefined} \end{cases}$$

We say that ϕ is PCF-definable if it definable by some (closed) PCF term.

Partial recursive functions

We will show that PCF-definable functions are precisely the partial recursive functions (as in Recursion Theory).

The set of **partial recursive functions** is the least set of partial functions $\phi : \mathbb{N}^n \rightharpoonup \mathbb{N}$ which:

- contains the *initial functions* (successor σ, predecessor π, zero ζ, and all projections Π_i^j),
- is closed under *function composition*,
- is closed under *primitive recursion*,
- is closed under *minimalisation*.

By the **Church-Turing thesis**, this class of functions represents the set of all **computable** functions. As a result, PCF is **Turing complete**.

Partial recursive functions explicitly

The set of **partial recursive functions** is the least set of partial functions $\mathcal{R} \subseteq \{\, \phi \mid \phi : \mathbb{N}^n \rightharpoonup \mathbb{N} \text{ for some } n \in \mathbb{N} \,\}$, for which:

- The functions σ, π, ζ and Π^i_j are in $\mathcal{R}$.
- If $\chi : \mathbb{N}^m \rightharpoonup \mathbb{N} \in \mathcal{R}$ and $\psi_i : \mathbb{N}^l \rightharpoonup \mathbb{N} \in \mathcal{R}$, for each $1 \leq i \leq m$, then $\phi : \mathbb{N}^l \rightharpoonup \mathbb{N} \in \mathcal{R}$, where:

$$\phi(n_1, \ldots, n_l) = \begin{cases} \chi(\psi_1(\vec{n}), \ldots, \psi_m(\vec{n})) & \text{if all } \psi_i(\vec{n}) \text{ are defined} \\ \text{undefined} & \text{otherwise} \end{cases}$$

Partial recursive functions explicitly (ctd)

The set of **partial recursive functions** is the least set of partial functions $\mathcal{R} \subseteq \{\, \phi \mid \phi : \mathbb{N}^n \rightharpoonup \mathbb{N} \text{ for some } n \in \mathbb{N} \,\}$, for which:

- If $\chi : \mathbb{N}^m \rightharpoonup \mathbb{N}$, $\psi : \mathbb{N}^{m+2} \rightharpoonup \mathbb{N} \in \mathcal{R}$ then $\phi : \mathbb{N}^{m+1} \rightharpoonup \mathbb{N} \in \mathcal{R}$, where:

$$\phi(k, n_1, \ldots, n_m) = \begin{cases} \chi(n_1, \ldots, n_m) & \text{if } k = 0 \\ \psi(\phi(k{-}1, n_1, \ldots, n_m), k{-}1, n_1, \ldots, n_m) & \text{if } k > 0 \\ & \text{and } \phi(k{-}1, n_1, \ldots, n_m) \text{ defined} \\ \text{undefined} & \text{otherwise} \end{cases}$$

- If $\chi : \mathbb{N}^{m+1} \rightharpoonup \mathbb{N} \in \mathcal{R}$ then $\phi : \mathbb{N}^m \rightharpoonup \mathbb{N} \in \mathcal{R}$, where:

$$\phi(n_1, \ldots, n_m) = \begin{cases} k & \text{if } k \text{ is the least number such that} \\ & \chi(k, n_1, \ldots, n_m) = 0, \text{ and } \chi(k', n_1, \ldots, n_m) \\ & \text{is defined for all } k' \leq k. \\ \text{undefined} & \text{otherwise (if no such } k \text{ exists)} \end{cases}$$

The Definability Theorem

Theorem. *A partial function $\phi : \mathbb{N}^m \rightharpoonup \mathbb{N}$ is recursive iff it is PCF-definable.*

For ($\Leftarrow$) we appeal to the **Church-Turing thesis**.
Let $\phi : \mathbb{N}^m \rightharpoonup \mathbb{N}$ be defined by a term M. Then M gives a procedure which, for any $\langle n_1, \ldots, n_m \rangle$, it returns $\phi(n_1, \ldots, n_m)$ if the latter is defined, otherwise it diverges: the procedure simply attempts to evaluate $M n_1 \ldots n_m$.
The procedure is computable, thus ϕ is recursive.

We do ($\Rightarrow$) as follows: we have already shown that the initial functions are PCF-definable, and we now show that PCF-definable functions are closed under composition, primitive recursion, and minimalisation.

Closure Under Composition

$$\frac{\chi : \mathbb{N}^m \rightharpoonup \mathbb{N} \in \mathcal{S} \quad \psi_1, \ldots, \psi_m : \mathbb{N}^l \rightharpoonup \mathbb{N} \in \mathcal{S}}{\phi : \mathbb{N}^l \rightharpoonup \mathbb{N} \in \mathcal{S}}$$

where $\phi(n_1, \ldots, n_l) = \chi(\psi_1(\vec{n}), \ldots, \psi_m(\vec{n}))$.

Suppose that χ is defined by M, and each ψ_i by N_i. Let:

$$N \equiv \lambda x_1 \ldots \lambda x_l . M(N_1 \vec{x}) \ldots (N_m \vec{x})$$

We get:

$$\frac{\dfrac{M(\psi_1(\vec{n})) \cdots (\psi_m(\vec{n})) \Downarrow \chi(\psi_1(\vec{n}), \ldots, \psi_m(\vec{n})) \quad N_i \vec{n} \Downarrow \psi_i(\vec{n})}{M(N_1 \vec{n}) \cdots (N_m \vec{n}) \Downarrow \chi(\psi_1(\vec{n}), \ldots, \psi_m(\vec{n}))}\ (Cong.)}{N n_1 \cdots n_l \Downarrow \chi(\psi_1(\vec{n}), \ldots, \psi_m(\vec{n}))}$$

Note that if $M(\psi_1(\vec{n})) \cdots (\psi_m(\vec{n})) \Uparrow$ then $N\vec{n} \Uparrow$ (by determinacy and congruence).
But what if some $N_i \vec{n} \Uparrow$?

Closure Under Composition (ctd)

For each $m \in \mathbb{N}$ we define a term of type $\underbrace{\mathtt{nat} \to \cdots \to \mathtt{nat}}_{m+1} \to \mathtt{nat}$

which checks if its first m inputs converge, and if so it returns its $m+1$-th input:

$$\mathtt{conv} \equiv \lambda x_1 \ldots \lambda x_m . \lambda y .\, \mathtt{cond}\ (\mathtt{zero?}\ (\mathtt{plus}\, x_1 (\mathtt{plus}\, x_2 (\ldots\ x_n))))\ y\ y$$

Let us now set:

$$N \equiv \lambda x_1 \ldots \lambda x_l .\, \mathtt{conv}\, (N_1 \vec{x}) \cdots (N_m \vec{x}) (M (N_1 \vec{x}) \ldots (N_m \vec{x}))$$

As before, we can show that $N n_1 \cdots n_l \Downarrow \chi(\psi_1(\vec{n}), \ldots, \psi_m(\vec{n}))$ if the latter is defined, and if $M(\psi_1(\vec{n})) \cdots (\psi_m(\vec{n})) \Uparrow$ or some $N_i \vec{n} \Uparrow$ then $N\vec{n} \Uparrow$.
Hence, N defines ϕ.

Closure Under Primitive Recursion

$$\frac{\chi : \mathbb{N}^m \rightharpoonup \mathbb{N} \in \mathcal{S} \quad \psi : \mathbb{N}^{m+2} \rightharpoonup \mathbb{N} \in \mathcal{S}}{\phi : \mathbb{N}^{m+1} \rightharpoonup \mathbb{N} \in \mathcal{S}}$$

where

$$\phi(k, n_1, \ldots, n_m) = \begin{cases} \chi(n_1, \ldots, n_m) & \text{if } k = 0 \\ \psi(\phi(k{-}1, n_1, \ldots, n_m), k{-}1, n_1, \ldots, n_m) & \text{otherwise} \end{cases}$$

Suppose that χ is defined by M and ψ by N. Let:

$$K \equiv \mathbf{Y}(\lambda f.\lambda y.\lambda x_1 \ldots \lambda x_m.\, K')$$

$$K' \equiv \texttt{cond}\ (\texttt{zero?}\ y)\ (M\vec{x})\ (\texttt{conv}\,(f(\texttt{pred}\,y)\vec{x})\,(N(f(\texttt{pred}\,y)\vec{x})\,(\texttt{pred}\,y)\,\vec{x})\,)$$

By induction on k it follows that K defines the required ϕ (Exercise).

Closure Under Minimalisation

$$\frac{\chi : \mathbb{N}^{m+1} \rightharpoonup \mathbb{N} \in \mathcal{S}}{\phi : \mathbb{N}^{m} \rightharpoonup \mathbb{N} \in \mathcal{S}}$$

where $\phi(n_1, \ldots, n_m) = k$, the least number such that $\chi(k, n_1, \ldots, n_m) = 0$.

Suppose that χ is defined by M. Let:

$$N \equiv \mathbf{Y}(\lambda f.\lambda y.\lambda x_1 \ldots \lambda x_m.\, \mathtt{cond}\ (\mathtt{zero?}\ (M y \vec{x}))\ y\ (f(\mathtt{succ}\, y)\vec{x}))$$

We claim that, for all $k \in \mathbb{N}$,

- if $\chi(k, n_1, \ldots, n_m) = 0$ then $N\, k\, n_1 \cdots n_m \Downarrow k$,
- if $\chi(k, n_1, \ldots, n_m)$ is not defined then $N\, k\, n_1 \cdots n_m \Uparrow$,
- otherwise, for all V, $N\, k\, n_1 \cdots n_m \Downarrow V$ iff $N(k{+}1)\, n_1 \cdots n_m \Downarrow V$.

Thus, taking $K \equiv N0$, we can see that K defines ϕ as above.

Closure Under Minimalisation (ctd)

$N \equiv \mathbf{Y}(\lambda f.\lambda y.\lambda x_1 \ldots \lambda x_m.\, \mathtt{cond}\ (\mathtt{zero?}\ (M y \vec{x}))\ y\ (f(\mathtt{succ}\, y)\vec{x}))$

We still need to show that, for all $k \in \mathbb{N}$,

- if $\chi(k, n_1, \ldots, n_m) = 0$ then $N\, k\, n_1 \cdots n_m \Downarrow k$,
- if $\chi(k, n_1, \ldots, n_m)$ is not defined then $N\, k\, n_1 \cdots n_m \Uparrow$,
- otherwise, for all V, $N k n_1 \cdots n_m \Downarrow V$ iff $N(k{+}1)\, n_1 \cdots n_m \Downarrow V$.

For the former, since M defines χ, we have that $M k \vec{n} \Downarrow 0$, so

$$\dfrac{\dfrac{\dfrac{M k \vec{n} \Downarrow 0 \qquad \overline{k \Downarrow k}}{\mathtt{cond}\ (\mathtt{zero?}\ (M k \vec{n}))\ k\ (N(\mathtt{succ}\, k)\vec{n}) \Downarrow k}}{(\lambda f.\lambda y.\lambda \vec{x}.\, \mathtt{cond}\ (\mathtt{zero?}\ (M y \vec{x}))\ y\ (f(\mathtt{succ}\, y)\vec{x}))N\, k\, \vec{n} \Downarrow k}\ (\text{m-app}_\lambda)}{N\, k\, \vec{n} \Downarrow k}\ (\text{m-app}_{\mathbf{Y}})$$

Closure Under Minimalisation (ctd)

$$N \equiv \mathbf{Y}(\lambda f.\lambda y.\lambda x_1 \ldots \lambda x_m.\, \mathtt{cond}\ (\mathtt{zero?}\ (My\vec{x}))\ y\ (f(\mathtt{succ}\, y)\vec{x}))$$

We still need to show that, for all $k \in \mathbb{N}$,

- if $\chi(k, n_1, \ldots, n_m) = 0$ then $N\, k\, n_1 \cdots n_m \Downarrow k$,
- if $\chi(k, n_1, \ldots, n_m)$ is not defined then $N\, k\, n_1 \cdots n_m \Uparrow$,
- otherwise, for all V, $Nkn_1 \cdots n_m \Downarrow V$ iff $N(k{+}1)\, n_1 \cdots n_m \Downarrow V$.

For the second, if $Nk\vec{n} \Downarrow V$ for some V, then by determinacy we get a contradiction:

$$\dfrac{\dfrac{\dfrac{Mk\vec{n} \Downarrow V' \qquad \vdots}{\mathtt{cond}\ (\mathtt{zero?}\ (Mk\vec{n}))\ k\ (N(\mathtt{succ}\, k)\vec{n}) \Downarrow k}}{(\lambda f.\lambda y.\lambda \vec{x}.\, \mathtt{cond}\ (\mathtt{zero?}\ (My\vec{x}))\ y\ (f(\mathtt{succ}\, y)\vec{x}))N\, k\, \vec{n} \Downarrow k}}{N\, k\, \vec{n} \Downarrow k}$$

Closure Under Minimalisation (ctd)

$$N \equiv \mathbf{Y}(\lambda f.\lambda y.\lambda x_1 \ldots \lambda x_m.\, \mathtt{cond}\ (\mathtt{zero?}\ (My\vec{x}))\ y\ (f(\mathtt{succ}\, y)\vec{x}))$$

We still need to show that, for all $k \in \mathbb{N}$,

- if $\chi(k, n_1, \ldots, n_m) = 0$ then $N\, k\, n_1 \cdots n_m \Downarrow k$,
- if $\chi(k, n_1, \ldots, n_m)$ is not defined then $N\, k\, n_1 \cdots n_m \Uparrow$,
- otherwise, for all V, $Nkn_1 \cdots n_m \Downarrow V$ iff $N(k{+}1)\, n_1 \cdots n_m \Downarrow V$.

For the latter, if $\chi(k, n_1, \ldots, n_m) = l + 1 > 0$ then, for any V, by determinacy we get:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{M k \vec{n} \Downarrow l+1} \qquad \dfrac{\vdots}{N(\mathtt{succ}\, k)\vec{n} \Downarrow V}}{\mathtt{cond}\ (\mathtt{zero?}\ (Mk\vec{n}))\ k\ (N(\mathtt{succ}\, k)\vec{n}) \Downarrow V}}{(\lambda f.\lambda y.\lambda \vec{x}.\, \mathtt{cond}\ (\mathtt{zero?}\ (My\vec{x}))\ y\ (f(\mathtt{succ}\, y)\vec{x}))N\, k\, \vec{n} \Downarrow V}}{N\, k\, \vec{n} \Downarrow V}$$

□

Corollaries

Corrolary. *There is no algorithm deciding whether PCF terms converge (i.e. evaluate to some value).*

Proof. If such an algorithm existed then, combined with the constructions of the previous theorem, it would yield an algorithm for deciding the **Halting problem**. □

Corrolary. *There is no algorithm deciding whether PCF terms evaluate to* 0.

Proof. The previous problem is undecidable for terms of type `nat`. But, for any $\vdash M : \mathtt{nat}$, we have that $M \Downarrow$ iff $\mathtt{conv}\, M\, 0 \Downarrow 0$. □

Equational theory of PCF

We next turn to the question:

When are two terms equivalent?

Possible answers:

- When they evaluate to the same value.
- When on all inputs they evaluate to same values.
- When on all inputs either they evaluate to same values or they both diverge.

Contextual Substitution

A **context** is a term with "holes", into which other terms can be plugged. E.g.

$$C[X] \equiv \lambda x.xXy$$

X is a called a **hole variable**. For a term M, we write $C[M]$ to mean the result of copying M for X "blindly" (i.e. not in a capture-avoiding manner) in $C[X]$. E.g:

$$\begin{aligned} C[z] &\equiv \lambda x.xzy \\ C[\lambda z.z] &\equiv \lambda x.x(\lambda z.z)y \\ C[\texttt{pred}] &\equiv \lambda x.x(\texttt{pred})y \\ C[xyy] &\equiv \lambda x.x(xyy)y \end{aligned}$$

Note how the variable x gets "captured" in the last contextual substitution.

Contextual equivalence

We define the relations of **contextual approximation** and **contextual equivalence** as follows. Let $\Gamma \vdash M : A$ and $\Gamma \vdash N : A$.

- We say that M *contextually approximates* N, written $\Gamma \vdash M \mathrel{\tilde{\sqsubseteq}} N : A$ or just $M \mathrel{\tilde{\sqsubseteq}} N$, if, for all contexts $C[X]$ such that $\vdash C[M] : \mathtt{nat}$ and $\vdash C[N] : \mathtt{nat}$,

$$C[M] \Downarrow 0 \implies C[N] \Downarrow 0$$

- We say that M and N are *contextually equivalent*, written $\Gamma \vdash M \cong N : A$ or just $M \cong N$, if $M \mathrel{\tilde{\sqsubseteq}} N$ and $N \mathrel{\tilde{\sqsubseteq}} M$. I.e. if, for all contexts $C[X]$ such that $\vdash C[M] : \mathtt{nat}$ and $\vdash C[N] : \mathtt{nat}$,

$$C[M] \Downarrow 0 \iff C[N] \Downarrow 0$$

Contextual equivalence is also called **observational equivalence**.

Examples

Here are some example (in)equivalences.

1. $\texttt{cond}\ x\ \texttt{t}\ \texttt{f}\ \cong?\ x$
2. $\texttt{cond}\ x\ \texttt{t}\ \texttt{t}\ \cong?\ \texttt{t}$
3. $\texttt{cond}\ x\ x\ \texttt{f}\ \cong?\ x$
4. $\texttt{cond}\ x\ x\ x\ \cong?\ x$

- $(\lambda x.\, \texttt{cond}\ (\texttt{zero}?\ x)\ 24\ 42)\, 42\ \cong\ 42$
- $\Omega_{\texttt{nat}\to\texttt{nat}}\ \cong\ \lambda x^{\texttt{nat}}.\Omega_{\texttt{nat}}$
- $x\ \ncong\ y$
- $\lambda x.\lambda y.\, \texttt{plus}\, x\, y\ \cong\ \lambda x.\lambda y.\, \texttt{plus}\, y\, x$

Equivalence contains evaluation

Lemma. *Suppose* $N \Downarrow V$. *Then, for any context* $C[X]$,

1. *if* $C[N] \Downarrow V'$ *then* $V' \equiv C'[N]$ *for some* $C'[X]$ *and* $C[V] \Downarrow C'[V]$;
2. *if* $C[V] \Downarrow V'$ *then* $V' \equiv C'[V]$ *for some* $C'[X]$ *and* $C[N] \Downarrow C'[N]$.

Proof. We show 2 (1 is shown in a similar fashion) by induction on the derivation of $C[V] \Downarrow V'$.
For the base case we have that $C[V]$ is a value, so

- either $C[X] \equiv \mathsf{c}$,
- or $C[X] \equiv \lambda y.C''[X]$,
- or $C[X] \equiv X$ and $V' \equiv V$.

The former case is straightforward: $C[N] \equiv \mathsf{c}$. For the middle one, we have that $C[N] \equiv \lambda y.C''[N]$, so the claim holds for $C'[X] \equiv \lambda y.C''[X]$. The latter follows from $N \Downarrow V'$.

Equivalence contains evaluation (ctd)

Lemma. *Suppose* $N \Downarrow V$. *Then, for any context* $C[X]$,

2. *if* $C[V] \Downarrow V'$ *then* $V' \equiv C'[V]$ *for some* $C'[X]$ *and* $C[N] \Downarrow C'[N]$.

Proof. We do induction on the derivation of $C[V] \Downarrow V'$. For the ind. step, $C[V]$ must be an application, so do case analysis on $C[V] \equiv C_1[V]C_2[V]$.

If $C_1[V]$ is not a value then $C[V] \Downarrow V'$ must be of the form:

$$\frac{C_1[V] \Downarrow V'' \quad V''C_2[V] \Downarrow V'}{C_1[V]C_2[V] \Downarrow V'}$$

Using the IH on $C_1[V]$, we get $V'' \equiv C_1'[V]$ and $C_1[N] \Downarrow C_1'[N]$.

Using the IH on $C_1'[V]C_2[V]$, we get $V' \equiv C'[V]$ and $C_1'[N]C_2[N] \Downarrow C'[N]$, thus

$$\frac{C_1[N] \Downarrow C_1'[N] \quad C_1'[N]C_2[N] \Downarrow C'[N]}{C_1[N]\, C_2[N] \Downarrow C'[N]}$$

Equivalence contains evaluation (ctd)

Lemma. *Suppose* $N \Downarrow V$. *Then, for any context* $C[X]$,

2. *if* $C[V] \Downarrow V'$ *then* $V' \equiv C'[V]$ *for some* $C'[X]$ *and* $C[N] \Downarrow C'[N]$.

Proof. We do induction on the derivation of $C[V] \Downarrow V'$. For the ind. step, $C[V]$ must be an application, so do case analysis on $C[V] \equiv C_1[V]C_2[V]$.

If $C_1[V] \equiv \mathsf{c} \not\equiv \mathbf{Y}$ then either $C_1[X] \equiv \mathsf{c}$, or $C_1[X] \equiv X$ and $V \equiv \mathsf{c}$.
In both cases, it must be:

$$\frac{C_2[V] \Downarrow U}{\mathsf{c}\, C_2[V] \Downarrow V'}$$

for some value U of base type. Thus, using the IH, we get either of:

$$\frac{C_2[N] \Downarrow U}{\mathsf{c}\, C_2[N] \Downarrow V'} \qquad \frac{N \Downarrow \mathsf{c} \quad \dfrac{C_2[N] \Downarrow U}{\mathsf{c}\, C_2[N] \Downarrow V'}}{N\, C_2[N] \Downarrow V'}$$

Equivalence contains evaluation (ctd)

Lemma. *Suppose* $N \Downarrow V$. *Then, for any context* $C[X]$,

2. *if* $C[V] \Downarrow V'$ *then* $V' \equiv C'[V]$ *for some* $C'[X]$ *and* $C[N] \Downarrow C'[N]$.

Proof. We do induction on the derivation of $C[V] \Downarrow V'$. For the ind. step, $C[V]$ must be an application, so do case analysis on $C[V] \equiv C_1[V]C_2[V]$.

If $C_1[V] \equiv \mathbf{Y}$ then the derivation must be:

$$\frac{C_2[V](\mathbf{Y}\, C_2[V]) \Downarrow V'}{\mathbf{Y}\, C_2[V] \Downarrow V'}$$

Using the IH on $C''[X] \equiv C_2[X](\mathbf{Y}\, C_2[X])$, we get that $V' \equiv C'[V]$ and $C_2[N](\mathbf{Y}\, C_2[N]) \Downarrow C'[N]$. Now, either $C_1[X] \equiv \mathbf{Y}$, or $C_1[X] \equiv X$ and $V \equiv \mathbf{Y}$. For each case use either of:

$$\frac{C_2[N](\mathbf{Y}\, C_2[N]) \Downarrow C'[N]}{\mathbf{Y}\, C_2[N] \Downarrow C'[N]} \qquad \frac{N \Downarrow \mathbf{Y} \qquad \dfrac{C_2[N](\mathbf{Y}\, C_2[N]) \Downarrow C'[N]}{\mathbf{Y}\, C_2[N] \Downarrow C'[N]}}{N\, C_2[N] \Downarrow C'[N]}$$

Equivalence contains evaluation (ctd)

Lemma. *Suppose* $N \Downarrow V$. *Then, for any context* $C[X]$,

2. *if* $C[V] \Downarrow V'$ *then* $V' \equiv C'[V]$ *for some* $C'[X]$ *and* $C[N] \Downarrow C'[N]$.

Proof. We do induction on the derivation of $C[V] \Downarrow V'$. For the ind. step, $C[V]$ must be an application, so do case analysis on $C[V] \equiv C_1[V]C_2[V]$.

Finally, if $C_1[V] \equiv \lambda y.C_1'[V]$ then we work similarly to the first subcase of the previous case. $\square$

Equivalence contains evaluation (at last)

Corrolary. *For any term N and value V, if $N \Downarrow V$ then $N \cong V$.*

Proof. For any context $C[X]$ such that $\vdash C[N] : \mathtt{nat}$, if $C[N] \Downarrow 0$ then, by the previous lemma, $C[V] \Downarrow 0$. And viceversa. □

For example,

- $(\lambda x.\, \mathtt{cond}\ (\mathtt{zero?}\ x)\ 24\ 42)\, 42 \ \cong\ 42$

Equivalence contains β-reduction

The meaning of applying a function to an argument is given by the relation:

$$(\lambda x.M)N \to_\beta M[N/x]$$

Although not mentioned explicitly, we have been using this in our evaluation rules.
We can show the following (Exercise).

Lemma. *For all terms M, N such that $\Gamma \vdash (\lambda x.M)N : A$, we have $(\lambda x.M)N \cong M[N/x]$.*

Further properties

- If $M \cong N$ then $C[M] \cong C[N]$ for any context $C[X]$.

  Proof. Take any context $C'[X]$ such that $C'[C[M]] \Downarrow 0$.
  Since $M \cong N$ (and examining the context $C'[C[X]]$) we get
  $C'[C[N]] \Downarrow 0$. And viceversa. □

- $M \cong N$ iff $\lambda x.M \cong \lambda x.N$.

  Proof. One direction ($\Rightarrow$) follows from the above.
  For the converse, let $\lambda x.M \equiv \lambda x.N$ and take $C[X]$ such that
  $C[M] \Downarrow 0$. By the previous lemma (β-reduction), we have that
  $M \cong (\lambda x.M)x$ and therefore $C[(\lambda x.M)x] \Downarrow 0$.

  But now $C[(\lambda x.M)x] \equiv C'[\lambda x.M]$, taking $C'[X] \equiv C[Xx]$.

  Thus, $\lambda x.M \cong \lambda x.N$ implies that $C'[\lambda x.N] \Downarrow 0$, i.e. $C[(\lambda x.N)x] \Downarrow 0$.
  Using the previous lemma, we get $C[N] \Downarrow 0$.
  And viceversa. □

Context Lemma

Contextual equivalence is highly combinatorial: it involves checking equality of evaluations under *all possible contexts* $C[X]$.

Actually, due to the purely functional behaviour of the language, not all these contexts are needed.
Let $\vdash M, N : A$, where $A = A_1 \to \cdots \to A_n \to A_b$.

- We say that M *applicatively approximates* N, written $M \mathrel{\underset{\sim}{\sqsubseteq}}^{\mathrm{app}} N$, if, for all closed terms $\vdash Q_i : A_i$ and values V,

$$MQ_1 \cdots Q_n \Downarrow V \implies NQ_1 \cdots Q_n \Downarrow V$$

We will show the following (known as *the Context Lemma*).

Lemma 1. *For all closed terms* $\vdash M, N : A$, $M \mathrel{\underset{\sim}{\sqsubseteq}} N \iff M \mathrel{\underset{\sim}{\sqsubseteq}}^{\mathrm{app}} N$.

Proof. One direction ($\Rightarrow$) is easy, since $MQ_1 \cdots Q_n \equiv C[M]$ with $C[X] \equiv XQ_1 \cdots Q_n$.

Proof of Context Lemma

For the other direction, we prove the following statement for all $n > 0$, by induction on n. We write $M \Downarrow_n V$ if $M \Downarrow V$ and the latter has derivation of size n.

- For all $\vdash M, N : A$, if $M \mathrel{\underset{\sim}{\sqsubset}}^{\mathrm{app}} N$ then, for any value V and context $C[X]$ such that $\vdash C[M], C[N] : A_b$, $C[M] \Downarrow_n V \implies C[N] \Downarrow V$.

Base case: Note that in this case we must have $C[M] \equiv V$, thus either $C[X] \equiv X$ and $M \equiv V$, or $C[X] \equiv V$. The latter case is obvious, while the former follows from $M \mathrel{\underset{\sim}{\sqsubset}}^{\mathrm{app}} N$.

Inductive step: We must have $C[X] \equiv C_0[X]C_1[X] \cdots C_m[X]$, some $m \geq 0$, and either of the following be the case.

1. $C[X] \equiv \mathsf{c}\, C_1[X] \cdots C_m[X]$, some constant c,
2. $C[X] \equiv (\lambda x^B.C_0'[X])C_1[X] \cdots C_m[X]$.
3. $C[X] \equiv X\, C_1[X] \cdots C_m[X]$,

Proof of Context Lemma (ctd)

Inductive case: There are several subcases:

1. $C[X] \equiv \mathsf{c}\, C_1[X] \cdots C_m[X]$, some constant c. We show $\mathsf{c} \equiv \mathtt{cond}$ and $\mathsf{c} \equiv \mathbf{Y}$ – the other cases are shown similarly.

For the former case, suppose $\mathtt{cond}\ C_1[M]\ C_2[M]\ C_3[M] \Downarrow_{n+1} V$.
By definition of $\Downarrow$, it must be that $C_1[M] \Downarrow_{n_1} V'$ for some $n_1 \leq n$ and $V' \in \{\mathtt{t}, \mathtt{f}\}$. Suppose WLOG that $V' \equiv \mathtt{t}$, in which case we also have $C_2[M] \Downarrow_{n_2} V$ for some $n_2 \leq n$.
By IH, we have $C_1[N] \Downarrow V'$ and $C_2[N] \Downarrow V$, and thus $C[N] \Downarrow V$.

Suppose now $\mathbf{Y}\, C_1[M] C_2[M] \cdots C_m[M] \Downarrow_{n+1} V$.

Observe that $(C_1[M](\mathbf{Y}\, C_1[M]))C_2[M] \cdots C_m[M] \Downarrow_{n'} V$, some $n' \leq n$, and thus, by the IH, applied on

$$C'[X] \equiv (C_1[X](\mathbf{Y}\, C_1[X]))C_2[X] \cdots C_m[X]$$

we obtain $(C_1[N](\mathbf{Y}\, C_1[N]))C_2[N] \cdots C_m[N] \Downarrow V$, which implies $\mathbf{Y}\, C_1[N] C_2[N] \cdots C_m[N] \Downarrow V$.

Proof of Context Lemma (ctd)

Inductive case: There are several subcases:

2. $C[X] \equiv (\lambda x.C'_0[X])\, C_1[X] \cdots C_m[X]$.

Suppose that $(\lambda x.C'_0[M])\, C_1[M] C_2[M] \cdots C_m[M] \Downarrow_{n+1} V$.

Observe that $(C'_0[M][C_1[M]/x]) C_2[M] \cdots C_m[M] \Downarrow_{n'} V$, some $n' \leq n$, and thus, by the IH, applied on

$$C'[X] \equiv (C'_0[X][C_1[X]/x]) C_2[X] \cdots C_m[X]$$

we obtain $(C'_0[N][C_1[N]/x]) C_2[N] \cdots C_m[N] \Downarrow V$, which implies $(\lambda x.C'_0[N])\, C_1[N] C_2[N] \cdots C_m[N] \Downarrow V$.

Proof of Context Lemma (ctd)

Inductive case: There are several subcases:

3. $C[X] \equiv X\, C_1[X] \cdots C_m[X]$.

In this case, $C[M] \equiv M\, C_1[M] C_2[M] \cdots C_m[M] \Downarrow_{n+1} V$.

Let $C'[X] \equiv M\, C_1[X] C_2[X] \cdots C_m[X]$. We have that $C'[M] \Downarrow_{n+1} V$ and, moreover, $C'[X]$ falls within one of the cases 1, 2 of our analysis.

Hence, $C'[N] \Downarrow V$, that is, $M\, C_1[N] C_2[N] \cdots C_m[N] \Downarrow V$.

But then, $M \underset{\sim}{\sqsubseteq}^{\mathrm{app}} N$ implies $N\, C_1[N] C_2[N] \cdots C_m[N] \Downarrow V$, which is as required. □

Applying the Context Lemma

1. $\texttt{cond}\ x\ \texttt{t}\ \texttt{f}\ \cong?\ x$
2. $\texttt{cond}\ x\ \texttt{t}\ \texttt{t}\ \cong?\ \texttt{t}$
3. $\texttt{cond}\ x\ x\ \texttt{f}\ \cong?\ x$
4. $\texttt{cond}\ x\ x\ x\ \cong?\ x$

- $\Omega_{\texttt{nat}\to\texttt{nat}}\ \cong\ \lambda x^{\texttt{nat}}.\Omega_{\texttt{nat}}$
- $\lambda x.\lambda y.\,\texttt{plus}\,x\,y\ \cong\ \lambda x.\lambda y.\,\texttt{plus}\,y\,x$

Models of PCF

By *models* of PCF we shall mean *denotational models* of PCF.

- This is justified by the fact that operational semantics is syntax-directed, whereas denotational models are "purer" (but take this with a pinch of salt . . .).

We saw that PCF terms can represent any recursive function.

Conversely, we will now like to see *every* PCF term as a "functional". That is, we want to construct a universe $\mathcal{M}$ such that, for each term $\Gamma \vdash M : A$, we can obtain some

$$[\![M]\!] : [\![\Gamma]\!] \to [\![A]\!]$$

– What are the specifications for such a model $\mathcal{M}$?

Full abstraction

The operational semantics induces a notion of term equivalence:

> Two PCF terms are equivalent if they cannot be distinguished within any PCF program

This allows us to say that:

- A model is **sound** if it *in*validates every PCF *in*equivalence.
- A model is **complete** if it validates every PCF equivalence.

A model is **fully abstract** if it is both sound and complete.

Full abstraction formally

Let $\mathcal{M}$ be a model of PCF with semantic translation $[\![\_]\!]$.
We say that:

- $\mathcal{M}$ is a **sound model** if, for all PCF terms M and N,

$$[\![M]\!] = [\![N]\!] \implies M \cong N$$

- $\mathcal{M}$ is a **complete model** if, for all PCF terms M and N,

$$M \cong N \implies [\![M]\!] = [\![N]\!]$$

A model $\mathcal{M}$ is **fully abstract** if it is both sound and complete:

$$M \cong N \iff [\![M]\!] = [\![N]\!]$$

Least fixpoints

In order to devise models of PCF, we need to account for fixpoints:

$$[\![\mathbf{Y}_A]\!] : [\![A \to A]\!] \to [\![A]\!]$$

Construct a semantic setting with canonical (*least*) fixpoints of functions.

This is analogous to ordinary recursive function definitions. For example, defining $\phi : \mathbb{N} \to \mathbb{N}$ as:

$$\phi(n) = \begin{cases} 1 & \text{if } n = 0 \\ n \times \phi(n-1) & \text{if } n > 0 \end{cases}$$

We mean the *least* function ϕ satisfying these specifications. Concretely, $\phi = \bigcup_{i \in \mathbb{N}} \phi_i$ (union of partial functions), where $\phi_0 = \emptyset$ and

$$\phi_{i+1} = \{(0,1)\} \cup \{\, (n, n \times m) \mid (n-1, m) \in \phi_i \,\}$$

Complete partial orders

- A **partially ordered set (poset)** is a structure $(P, \sqsubseteq)$ where $\sqsubseteq$ is a relation on the set P such that, for all $x, y \in P$,

Reflexivity	*Transitivity*	*Anti-symmetry*
$x \sqsubseteq x$	$x \sqsubseteq y \wedge y \sqsubseteq z \implies x \sqsubseteq z$	$x \sqsubseteq y \wedge y \sqsubseteq x \implies x = y$

- $\bot \in P$ is a **least element** of a poset P if $\bot \sqsubseteq x$ for all $x \in P$.

- Given $S \subseteq P$, an $x \in P$ is an **upper bound** of S if $y \sqsubseteq x$ for all $y \in S$. x is a **least upper bound (lub)** if, additionally, $x \sqsubseteq x'$ for any upper bound x' of S.

- A **chain** (or ω-*chain*) in P is a sequence of elements $(x_i)_{i \in \mathbb{N}}$ of P such that $x_i \sqsubseteq x_{i+1}$ for all $i \in \mathbb{N}$.
  A lub on $(x_i)_{i \in \mathbb{N}}$ is a lub of the set $\{x_i \mid i \in \mathbb{N}\}$. We write this as $\bigsqcup_{i \in \mathbb{N}} x_i$.

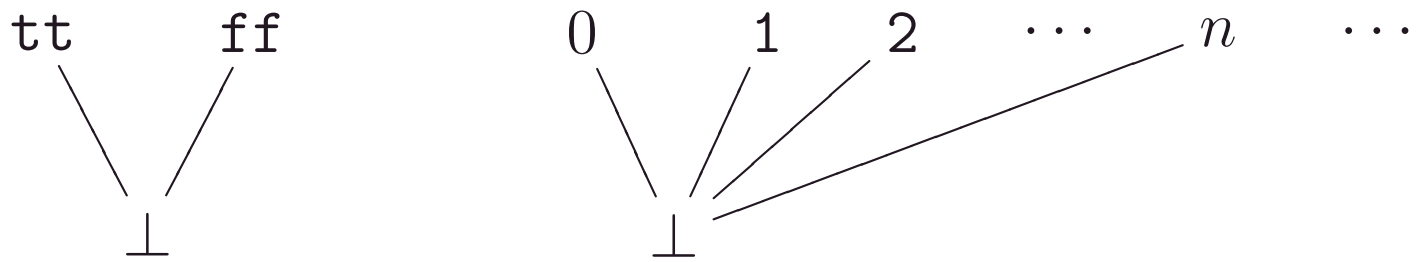
A **complete partial order (cpo)** is a poset $(D, \sqsubseteq)$ which has a least element, and a lub for every chain.

Examples

Flat cpo's Given a set X, we can form a cpo $X_\bot$ adjoining an element $\bot \notin X$ and defining the order by:

$$x \sqsubseteq y \iff x = \bot \vee x = y$$

E.g. the cpos $\mathbb{B}_\bot$ (where $\mathbb{B} = \{\mathtt{tt}, \mathtt{ff}\}$) and $\mathbb{N}_\bot$:



Powersets Given a set X, the powerset $\mathcal{P}(X)$, ordered by inclusion, is a cpo. In particular, $\bot = \emptyset$ and

$$\bigsqcup_{i \in \omega} X_i = \bigcup_{i \in \omega} X_i$$

Examples (ctd)

Streams Let Σ be some set (*alphabet*), and Σ^∞ be the set of all (finite and infinite) countable sequences of elements of Σ. Σ^∞ is a cpo, where sequences are ordered by *prefix*:

$$x \sqsubseteq y \iff x = y \ \vee \ \exists z.\, xz = y$$

Least element is the empty sequence, while if $x_0 \sqsubseteq x_1 \sqsubseteq x_2 \cdots$ is a chain then $\bigsqcup_i x_i$ is ...

Products Let D, E be cpo's. Their cartesian product $D \times E$ with the *componentwise* ordering:

$$(x, y) \sqsubseteq (x', y') \iff x \sqsubseteq x' \wedge y \sqsubseteq y'$$

is a cpo. Least element and lub's are also determined componentwise:

$$\bot_{D\times E} = (\bot_D, \bot_E) \qquad \bigsqcup_{i\in\mathbb{N}} (x_i, y_i) = (\bigsqcup_{i\in\mathbb{N}} x_i, \bigsqcup_{i\in\mathbb{N}} y_i)$$

Continuous functions

We consider an appropriate notion of function between cpo's.
Let D, E be cpo's. Consider a function $f : D \to E$.

- f is **monotonic** if, for all $x, y \in D$, $x \sqsubseteq y \implies f(x) \sqsubseteq f(y)$.
- f is **continuous** if it is monotonic and, additionally, for all chains $(x_i)_{i\in\mathbb{N}}$ in D,
$$f(\bigsqcup_{i\in\mathbb{N}} x_i) = \bigsqcup_{i\in\mathbb{N}} f(x_i)\,.$$

We write $D \Rightarrow E$ for the the set of continuous functions $f : D \to E$.
This is also a cpo, ordered componentwise:

$$f \sqsubseteq g \iff \forall x \in D.\, f(x) \sqsubseteq f(y)$$

The least element of $D \Rightarrow E$ is the function $\bot_{D\Rightarrow E} = x \mapsto \bot_E$.

Lub's are computed componentwise: $(\bigsqcup_{i\in\mathbb{N}} f_i)(x) = \bigsqcup_{i\in\mathbb{N}} f_i(x)$.

Notes

- We usually refer to a cpo $(D, \sqsubseteq)$ simply as D. We may use subscripts, like in $\sqsubseteq_D$ or $\bot_D$, to specify the cpo structure of D.
- Products between cpo's extend to finite products $D_1 \times \cdots \times D_n$ in a straightforward manner.
- Note that if $f : D \to E$ is monotonic and $(x_i)_{i \in \mathbb{N}}$ is a chain in D then $(f(x_i))_{i \in \mathbb{N}}$ is a chain in E.
  Moreover, $f(x_i) \sqsubseteq f(\bigsqcup_{i \in \mathbb{N}} x_i)$, for each i, and therefore

$$\bigsqcup_{i \in \mathbb{N}} f(x_i) \sqsubseteq f(\bigsqcup_{i \in \mathbb{N}} x_i)$$

The Fixpoint Theorem

A **fixpoint** of a function $f : D \to D$ is an $x \in D$ such that $f(x) = x$.
x is a **least fixpoint** if, additionally, $x \sqsubseteq x'$ for every fixpoint x' of f.

Theorem. *Let D be a cpo and $f : D \to D$ a continuous function. Then f has a least fixpoint $\mathtt{lfp}(f)$, which is defined explicitly by:*

$$\mathtt{lfp}(f) = \bigsqcup_{i \in \mathbb{N}} f^i(\bot) \qquad (\textit{where } f^0(\bot) = \bot,\ f^{i+1}(\bot) = f(f^i(\bot)))$$

Proof. We first show that $(f^i(\bot))_{i \in \mathbb{N}}$ is a chain, i.e. $f^i(\bot) \sqsubseteq f^{i+1}(\bot)$, by induction on i: $\bot \sqsubseteq f(\bot)$ by leastness, and $f^i(\bot) \sqsubseteq f^{i+1}(\bot)$ implies $f^{i+1}(\bot) \sqsubseteq f^{i+2}(\bot)$ by monotonicity.

Next we show that $\mathtt{lfp}(f)$ is a fixpoint of f. By continuity of f:

$$f(\bigsqcup_{i \in \mathbb{N}} f^i(\bot)) = \bigsqcup_{i \in \mathbb{N}} f(f^i(\bot)) = \bigsqcup_{i \in \mathbb{N}} f^{i+1}(\bot) \overset{(Ex.)}{=} \bigsqcup_{i \in \mathbb{N}} f^i(\bot)$$

The Fixpoint Theorem

A **fixpoint** of a function $f : D \to D$ is an $x \in D$ such that $f(x) = x$.
x is a **least fixpoint** if, additionally, $x \sqsubseteq x'$ for every fixpoint x' of f.

Theorem. *Let D be a cpo and $f : D \to D$ a continuous function. Then f has a least fixpoint $\mathtt{lfp}(f)$, which is defined explicitly by:*

$$\mathtt{lfp}(f) = \bigsqcup_{i \in \mathbb{N}} f^i(\bot) \qquad (\textit{where } f^0(\bot) = \bot,\ f^{i+1}(\bot) = f(f^i(\bot)))$$

Proof. Finally, suppose that x is a fixpoint of f. We show by induction on i that $f^i(\bot) \sqsubseteq x$ for all $i \in \mathbb{N}$.
For $i = 0$ this is clear, and if $f^i(\bot) \sqsubseteq x$ then:

$$f^{i+1}(\bot) = f(f^i(\bot)) \sqsubseteq f(x) = x$$

Thus, x is an upper bound for $(f^i(\bot))_{i \in \mathbb{N}}$ and so $\bigsqcup_{i \in \mathbb{N}} f^i(\bot) \sqsubseteq x$. $\square$

Exercises

1. Complete the primitive recursion case of the proof of the Definability Theorem.

2. Without using the given Corollaries, show that there is no algorithm deciding whether closed PCF terms of type `bool` converge.

3. Show that the following are equivalent definitions of $\underset{\sim}{\sqsubseteq}$.
   - For all $n \in \mathbb{N}$ and all contexts $C[X]$ such that $\vdash C[M], C[N] : \mathtt{nat}$, $C[M] \Downarrow n \implies C[N] \Downarrow n$.
   - For all contexts $C[X]$ such that $\vdash C[M], C[N] : \mathtt{nat}$, $C[M] \Downarrow \implies C[N] \Downarrow$.

4. Prove that β-reduction is contained in contextual equivalence.

5. Extend $\underset{\sim}{\sqsubseteq}^{\mathrm{app}}$ to open terms as follows.
   For each $\Gamma \vdash M, N : A$, where $\Gamma = x_1 : B_1, \ldots, x_n : B_n$, let $\Gamma \vdash M \underset{\sim}{\sqsubseteq}^{\mathrm{app}} N : A$ if $\lambda x_1^{A_1} \ldots x_n^{A_n}.M \underset{\sim}{\sqsubseteq}^{\mathrm{app}} \lambda x_1^{A_1} \ldots x_n^{A_n}.N$.
   Using the Context Lemma, show that $\underset{\sim}{\sqsubseteq} = \underset{\sim}{\sqsubseteq}^{\mathrm{app}}$.