Topic 5

PCF
PCF syntax

Types

\[ \tau ::= \text{nat} \mid \text{bool} \mid \tau \rightarrow \tau \]

Expressions

\[ M ::= 0 \mid \text{succ}(M) \mid \text{pred}(M) \]

\[ \mid \text{true} \mid \text{false} \mid \text{zero}(M) \]

\[ \mid x \mid \text{if } M \text{ then } M \text{ else } M \]

\[ \mid \text{fn } x : \tau . M \mid M \_M \mid \text{fix}(M) \]

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

Technicality: We identify expressions up to \( \alpha \)-conversion of bound variables (created by the \text{fn} expression-former): by definition a PCF term is an \( \alpha \)-equivalence class of expressions.
PCF typing relation (sample rules)

\[
\begin{align*}
(\text{fn}) \\
\Gamma \vdash M : \tau' \\
\Gamma \vdash \text{fn } x : \tau \cdot M : \tau \rightarrow \tau' \\
\text{if } x \notin \text{dom}(\Gamma)
\end{align*}
\]
PCF typing relation (sample rules)

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

\[
\frac{\Gamma \vdash M_1 : \tau \to \tau' \quad \Gamma \vdash M_2 : \tau}{\Gamma \vdash M_1 \, M_2 : \tau'}
\]

\[
\frac{\Gamma \vdash M : \tau \to \tau}{\Gamma \vdash \text{fix}(M) : \tau}
\]
Partial recursive functions in PCF

- Primitive recursion.

\[
\begin{align*}
   h(x, 0) &= f(x) \\
   h(x, y + 1) &= g(x, y, h(x, y))
\end{align*}
\]

\[H : \text{nat} \rightarrow \text{nat} \rightarrow \text{nat}\]

\[
H \; x \; y = \begin{cases} 
    F(x) & \text{if } \text{zero}(y) \\
    G \; x \; \text{pred}(y) \; (H \; x \; \text{pred}(y)) & \text{else}
\end{cases}
\]

\[
H = \text{fix}(\lambda h. \lambda x. \lambda y. \text{if } \text{zero}(y) \text{ then } F(x) \text{ else } G \; x \; \text{pred}(y))
\]
Partial recursive functions in PCF

- Primitive recursion.

\[
\begin{aligned}
  h(x, 0) &= f(x) \\
  h(x, y + 1) &= g(x, y, h(x, y))
\end{aligned}
\]

- Minimisation.

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

\[
\begin{aligned}
  H x y &= \text{if } \overline{\text{zero}}(k x y) \text{ then } y \\
  &\text{else } H x (\text{succ } y)
\end{aligned}
\]

\[
H > \text{fix}(\lambda h. H x.)
\]

\[
m > \text{fix}_c \cdot H x 0
\]
PCF evaluation relation

takes the form

\[ M \downarrow_{\tau} V \]

where

- \( \tau \) is a PCF type
- \( M, V \in \text{PCF}_{\tau} \) are closed PCF terms of type \( \tau \)
- \( V \) is a value,

\[ V ::= 0 \mid \text{succ}(V) \mid \text{true} \mid \text{false} \mid \text{fn } x : \tau . M. \]
PCF evaluation (sample rules)

\[
\begin{align*}
(\downarrow_{\text{val}}) & \quad V \downarrow_{\tau} V \quad (V \text{ a value of type } \tau) \\
(\downarrow_{\text{cbn}}) & \quad \frac{M_1 \downarrow_{\tau \rightarrow \tau'} \text{ fn } x : \tau . M_1' \quad M_1'[M_2/x] \downarrow_{\tau'} V}{M_1 M_2 \downarrow_{\tau'} V}
\end{align*}
\]
PCF evaluation (sample rules)

\[
\begin{align*}
(\downarrow_{\text{val}}) & \quad V \downarrow_{\tau} V \quad (V \text{ a value of type } \tau) \\
(\downarrow_{\text{cbn}}) & \quad \frac{M_1 \downarrow_{\tau \to \tau'} \text{ fn } x : \tau . M'_1 \quad M'_1[M_2/x] \downarrow_{\tau'} V}{M_1 \quad M_2 \downarrow_{\tau'} V} \\
(\downarrow_{\text{fix}}) & \quad \frac{M \text{ fix}(M) \downarrow_{\tau} V}{\text{ fix}(M) \downarrow_{\tau} V}
\end{align*}
\]
There is no \( V \) s.t. \( \forall x \ (\text{fn}(x; 2.2) \downarrow \top) \)

\[\text{fn}(x; 2.2 \downarrow) \quad x \quad \text{id} \quad \text{id} \quad \text{fn}(x; 2.2) \quad \top \]

\[\text{fn}(x; 2.2) \quad \text{fn}(x; 2.2) \quad x \quad \text{fn}(\text{fn}(x; 2.2); 2) \quad \top \]

\[\text{fn}(\text{fn}(x; 2.2)) \quad \text{fn}(\text{fn}(x; 2.2); 2) \quad \top \]

\[\text{fn}(\text{fn}(x; 2.2)) \quad \text{fn}(\text{fn}(x; 2.2); 2) \quad \top \]
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.
Contextual equivalence of PCF terms

Given PCF terms $M_1, M_2$, PCF type $\tau$, and a type environment $\Gamma$, the relation $\Gamma \vdash M_1 \simeq_{\text{ctx}} M_2 : \tau$ is defined to hold iff

- Both the typings $\Gamma \vdash M_1 : \tau$ and $\Gamma \vdash M_2 : \tau$ hold.
- For all PCF contexts $C$ for which $C[M_1]$ and $C[M_2]$ are closed terms of type $\gamma$, where $\gamma = \text{nat}$ or $\gamma = \text{bool}$, and for all values $V : \gamma$,

$$C[M_1] \Downarrow_\gamma V \iff C[M_2] \Downarrow_\gamma V.$$
PCF denotational semantics — aims

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

\[
\llbracket \text{nat} \rrbracket = \downarrow 0 \downarrow 1 \cdots \downarrow n \cdots
\]

\[
\llbracket \text{bool} \rrbracket = \text{ht} \text{ff}
\]

\[
\llbracket \text{nat} \rightarrow \text{nat} \rrbracket = (\prod \text{nat} \rightarrow \prod \text{nat})
\]
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.

\[
\frac{x_1 : \tau_1, x_2 : \tau_2, \ldots, x_n : \tau_n \vdash M : \tau}{[\tau_1] \times [\tau_2] \times \cdots \times [\tau_n] \rightarrow [\tau]}\]
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: $[M] = [M'] \Rightarrow [C[M]] = [C[M']]$.
- Soundness.
  For any type $\tau$, $M \Downarrow^\tau V \Rightarrow [M] = [V]$.
- Adequacy.
  For $\tau = \text{bool}$ or $\text{nat}$, $[M] = [V] \in [\tau] \implies M \Downarrow^\tau V$. 
**Theorem.** For all types $\tau$ and closed terms $M_1, M_2 \in \text{PCF}_\tau$, if $\llbracket M_1 \rrbracket$ and $\llbracket M_2 \rrbracket$ are equal elements of the domain $\llbracket \tau \rrbracket$, then $M_1 \simeq_{\text{ctx}} M_2 : \tau$.

$$C\llbracket M_1 \rrbracket \downarrow \nu \implies \llbracket C(M_1) \rrbracket = \llbracket \nu \rrbracket$$

$$\implies \llbracket C(M_2) \rrbracket \downarrow = \llbracket \nu \rrbracket$$

$$\implies C\llbracket M_2 \rrbracket \downarrow \nu$$
Theorem. For all types $\tau$ and closed terms $M_1, M_2 \in \text{PCF}_\tau$, if $\llbracket M_1 \rrbracket$ and $\llbracket M_2 \rrbracket$ are equal elements of the domain $\llbracket \tau \rrbracket$, then $M_1 \cong_{\text{ctx}} M_2 : \tau$.

Proof.

$$C[M_1] \downarrow_{\text{nat}} V \Rightarrow \llbracket C[M_1] \rrbracket = \llbracket V \rrbracket \quad \text{(soundness)}$$

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

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

and symmetrically. \hfill \square
Proof principle

To prove

\[ M_1 \simeq_{\text{ctx}} M_2 : \tau \]

it suffices to establish

\[ [M_1] = [M_2] \text{ in } [\tau] \]
Proof principle

To prove

\[ M_1 \simeq_{\text{ctx}} M_2 : \tau \]

it suffices to establish

\[ \llbracket M_1 \rrbracket = \llbracket M_2 \rrbracket \text{ in } \llbracket \tau \rrbracket \]

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