Lecture 5

PCF
Definition: A PCF term is an $\alpha$-equivalence class of expressions.

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.

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

Expressions

$$\forall x \in \mathbb{V} \quad (W)(\, \text{fix} \mid W \cdot \bot \mid x \mid W \text{ if then else } W \mid x \mid W \text{ true false} \mid W \text{ pred succ} \mid 0) = \cdot : W$$

Types
\[
\{ \bot : \mathcal{W} \mid \mathcal{W} \} \overset{\text{def}}{=} \text{PCF}
\]

\( \mathcal{W} \) is a type environment, i.e., a finite partial function mapping variables to types (whose domain of definition is denoted \( \text{dom}(\mathcal{W}) \)).

\( \bot : \mathcal{W} \overset{\Delta}{\rightarrow} \emptyset \) holds.

\( \mathcal{W} \) is a term

\( \bot : \mathcal{W} \) means \( \mathcal{W} \) is closed and \( \bot : \mathcal{W} \) holds.

\( \text{PCF-typing relation, } \mathcal{W} \overset{\Delta}{\rightarrow} \bot \)

Notation:

- \( \bot \) is a type.
- \( \mathcal{W} \) is a term
- \( \text{dom}(\bot) \) is a type environment, i.e., a finite partial function mapping variables to types (whose domain of definition is denoted \( \text{dom}(\mathcal{W}) \)).
PCF typing relation (sample rules)

\[
\begin{align*}
\Gamma &\vdash \text{fn} & \\
\Gamma &\vdash \text{fix} & \\
\Gamma &\vdash M : \tau' & \\
\Gamma &\vdash M_1 : \tau & \\
\Gamma &\vdash M_2 : \tau & \\
\Gamma &\vdash \text{fix}(M) : \tau & \\
\end{align*}
\]
Partial recursive functions in PCF

- Primitive recursion.

\[ h(x) = f(x) \]
\[ h(x + 1) = g(x, h(x)) \]

- Minimisation.

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

\[ m = (x)m \]

\[ ((h, x)y)(h, x)y = (1 + h, x)y \]
\[ (x)f = (0, x)y \]

- Primitive recursion.

\[ 0 = (h, x)y \]

Partial Recursive Functions in PCF
PCF evaluation relation takes the form

\[ M \Downarrow \tau \Downarrow V \in \text{PCF} \]

where

- \( \tau \) is a PCF type
- \( M \Downarrow \) is a PCF term of type \( \tau \)
- \( V \) is a value
- \( V ::= \bot \mid \text{true} \mid \text{false} \mid \text{fn } x : \tau . M \)

where

\[ \Lambda \Downarrow \cdot \mapsto M \]

takes the form
PCF evaluation (sample rules)

\[
\lambda \vdash (\forall x y. M) \quad (\forall x y. \vdash)
\]

\[
\lambda \vdash (\forall x y. \forall M \forall W)
\]

\[
\lambda \vdash [x \forall M \forall W] \quad (\forall M \forall W. \perp : x \text{ un} \leftarrow \vdash \forall M \forall W)
\]

\[
\lambda \vdash (\forall M \forall W)
\]

(\text{a value of type } \perp)

\[
\lambda \vdash \perp \quad (\forall \perp. \vdash)
\]
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.
Given PCF terms $M_1 \equiv M_2$, PCF type $\tau$, and a type environment $\Gamma$, the relation $\Gamma \vdash M_1 \equiv ctx \Gamma_2$ 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 $\gamma = \mathtt{nat}$ or $\gamma = \mathtt{bool}$, and for all values $V : \gamma$, $C[M_1] \Downarrow \gamma V \iff C[M_2] \Downarrow \gamma V$.

\[ \forall \gamma : \text{type} \quad \forall M_1, M_2 \quad M_1 \equiv M_2 \Rightarrow \bigtriangleup \Gamma_2 \bigtriangleup M_1 \bigtriangleup M_2 \bigtriangleup \Gamma \]
PCF denotational semantics — aims

• PCF types
\[ \tau \rightarrow \text{domains} \]

• Closed PCF terms
\[ M : \tau \rightarrow \text{elements} \in \lbrack \tau \rbrack \]

Denotations of open terms will be continuous functions.

• Compositionality

\[ \lbrack \Lambda \rbrack = \lbrack W \rbrack \iff \Lambda \uparrow \downarrow W \]

In particular:

\[ \lbrack W.C \rbrack = \lbrack W \rbrack \IFF \lbrack W \rbrack = \lbrack W \rbrack \]

• Soundness

For any type \( \tau \),

\[ M \downarrow \tau : V \Rightarrow \lbrack M \rbrack = \lbrack V \rbrack \]

• Adequacy

For \( \tau = \text{bool or nat} \),

\[ \lbrack M \rbrack = \lbrack V \rbrack \Rightarrow M \downarrow \tau V \]

• PCF types
\[ \lbrack \bot \rbrack \Downarrow \text{domains} \]

PCF denotational semantics — aims
Theorem. For all types $\tau$ and closed terms $M_1 \equiv M_2 \in \text{PCF}_\tau$, if $\lbrack\lbrack M_1\rbrack\rbrack$ and $\lbrack\lbrack M_2\rbrack\rbrack$ are equal elements of the domain $\lbrack\lbrack \tau \rbrack\rbrack$, then $M_1 \sim_{\text{ctx}} M_2$.

Proof. $\exists \lbrack\lbrack M_1 \rbrack\rbrack : \tau. \text{ctx}\ W_2 \equiv W_1 \text{ and closed terms } W_1, W_2 \in \text{PCF}_\tau$. For all types $\tau$ and closed terms $W_1, W_2 \in \text{PCF}_\tau$, if $\lbrack\lbrack W_1 \rbrack\rbrack \equiv \lbrack\lbrack W_2 \rbrack\rbrack$, then $\lbrack\lbrack \Lambda \rbrack\rbrack = \lbrack\lbrack \lbrack\lbrack W_1 \rbrack\rbrack \cdot \rbrack\rbrack \rbrack \iff \lbrack\lbrack W_2 \rbrack\rbrack = \lbrack\lbrack M_2 \rbrack\rbrack$. 

\[ (\lbrack\lbrack M_1 \rbrack\rbrack = \lbrack\lbrack M_2 \rbrack\rbrack \text{ on compositionality} \implies \lbrack\lbrack C M_2 \rbrack\rbrack = \lbrack\lbrack V \rbrack\rbrack \text{ (soundness)} \implies \lbrack\lbrack C M_2 \rbrack\rbrack = \lbrack\lbrack V \rbrack\rbrack \text{ (compositionality on } \lbrack\lbrack M_1 \rbrack\rbrack = \lbrack\lbrack M_2 \rbrack\rbrack) \implies \lbrack\lbrack C M_2 \rbrack\rbrack = \lbrack\lbrack V \rbrack\rbrack \text{ (soundness) \implies } \lbrack\lbrack W_2 \rbrack\rbrack = \lbrack\lbrack M_2 \rbrack\rbrack \rbrack. \]
Proof principle

To prove $M_1 \sim = \text{ctx} M_2$, it suffices to establish $\llbracket \text{in} \rrbracket M_1 \ggbracket = \llbracket \text{in} \rrbracket M_2$.

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

\[ \llbracket \text{in} \rrbracket M_2 \equiv^\text{ctx} M_1 \]

It suffices to establish

\[ \text{To prove} \]

Proof Principle