Lecture 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, \( \Gamma \vdash M : \tau \)

- \( \Gamma \) is a type environment, \( i.e. \) a finite partial function mapping variables to types (whose domain of definition is denoted \( \text{dom}(\Gamma) \))
- \( M \) is a term
- \( \tau \) is a type.

Notation:

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

PCF\( _\tau \) def = \{ M \mid M : \tau \} \)
Partial recursive functions in PCF

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

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

PCF evaluation relation

\[
M \Downarrow \tau V
\]

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)

\[
(V \Downarrow \text{val}) \quad V \Downarrow \tau V \quad (V \text{ a value of type } \tau)
\]

\[
(V \Downarrow \text{cbn}) \quad M_1 \Downarrow \tau, x : \tau . M'_1 \quad M'_1[M_2/x] \Downarrow \tau, V \quad \frac{}{M_1 M_2 \Downarrow \tau, V}
\]

\[
(V \Downarrow \text{fix}) \quad M \text{fix}(M) \Downarrow \tau V \quad \frac{}{\text{fix}(M) \Downarrow \tau V}
\]

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 \( M_1 \) and \( M_2 \), PCF types \( \tau \), and an environment \( \Gamma \), the relation

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

is defined to hold if both typings \( \Gamma \vdash \mathcal{C}[M_1] : \tau \) and \( \Gamma \vdash \mathcal{C}[M_2] : \tau \) hold.

For all PCF contexts \( C \) for which \( \mathcal{C}[M_1] \) and \( \mathcal{C}[M_2] \) are closed terms of type \( \tau \), where \( \tau = \text{nat} \) or \( \tau = \text{bool} \), and for all values \( V \):

\[ C[M_1] \xrightarrow{\text{ctx}} C[M_2] : \tau. \]

Theorem. For all types \( \tau \) and closed terms \( M_1, M_2 \in \text{PCF}_r \), if \( \mathcal{C}[M_1] \xrightarrow{\text{ctx}} \mathcal{C}[M_2] : \tau \), then:

\[ M_1 \xi \text{ctx} M_2 : \tau. \]

Proof. It suffices to establish \( \mathcal{C}[M_1] = \mathcal{C}[M_2] \) in \( \mathcal{C}[\xi] \).

Compositionality.

For any type \( \tau \), \( M \vdash V \Rightarrow [M] = [V] \).

Soundness.

In particular, if \( [M] = [V] \Rightarrow \mathcal{C}[M] = \mathcal{C}[V] \).

Adequacy.

For any type \( \tau \), \( M \vdash \mathcal{C}[M] : \tau \Rightarrow \mathcal{C}[M] = [M] : \tau \).

Proof principle.

To prove it suffices to establish \( M_1 \xi \text{ctx} M_2 : \tau \).

Given PCF terms \( M_1, M_2 \), PCF type \( \tau \), and a type environment \( \Gamma \), the relation \( \Gamma \vdash \mathcal{C}[M_1] = \mathcal{C}[M_2] : \tau \) is defined to hold if:

\[ \mathcal{C}[M_1] \xrightarrow{\text{ctx}} \mathcal{C}[M_2] : \tau. \]

Closed PCF terms \( M : \tau \rightarrow \text{domains} \in \mathcal{C}[\tau] \).

Denotations of open terms will be continuous functions.

PCF denotational semantics — aims

and symmetrically.

\[ \mathcal{C}[M_2] \xrightarrow{\text{ctx}} \mathcal{C}[M_1] : \tau \]

(adequacy)

(compositionality on \( \mathcal{C}[\xi] = \mathcal{C}[\xi] \))

(soundness)

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