Topic 5

PCF
Types

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

Types

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

Expressions

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

Types

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

Expressions

\[ M ::= 0 \mid \text{succ}(M) \mid \text{pred}(M) \]
\[ \mid \text{true} \mid \text{false} \mid \text{zero}(M) \]
PCF syntax

Types

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

Expressions

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

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

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

Types

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

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 \( x \in \mathcal{V} \), 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.
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.

$\text{PCF}_\tau \overset{\text{def}}{=} \{ M \mid M : \tau \}$. 
PCF typing relation (sample rules)

\[\begin{align*}
&\text{(:fn)} & 
\frac{\Gamma[x \mapsto \tau] \vdash M : \tau'}{
\Gamma \vdash \text{fn } x : \tau . M : \tau \to \tau'} 
\quad \text{if } x \notin \text{dom}(\Gamma)
\end{align*}\]
PCF typing relation (sample rules)

\[
\begin{align*}
(\text{:fn}) & \quad \frac{\Gamma[x \mapsto \tau] \vdash M : \tau'}{
\Gamma \vdash \text{fn} \ x : \tau \cdot M : \tau \rightarrow \tau'} \quad \text{if } x \notin \text{dom}(\Gamma) \\
(\text{:app}) & \quad \frac{\Gamma \vdash M_1 : \tau \rightarrow \tau' \quad \Gamma \vdash M_2 : \tau}{
\Gamma \vdash M_1 \ M_2 : \tau'}
\end{align*}
\]
PCF typing relation (sample rules)

\[\text{(\(\because\text{fn}\)) \quad \frac{\Gamma[x \mapsto \tau] \vdash M : \tau'}{\Gamma \vdash \text{fn} x : \tau . M : \tau \rightarrow \tau'} \quad \text{if } x \notin \text{dom}(\Gamma)}\]

\[\text{(\(\because\text{app}\)) \quad \frac{\Gamma \vdash M_1 : \tau \rightarrow \tau' \quad \Gamma \vdash M_2 : \tau}{\Gamma \vdash M_1 M_2 : \tau'}\]

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

- Primitive recursion.

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

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)

\[(\downarrow_{\text{val}}) \quad V \downarrow_\tau V \quad (V \text{ a value of type } \tau)\]
PCF evaluation (sample rules)

\( (\downarrow_{\text{val}}) \ V \downarrow_{\tau} V \quad (V \text{ a value of type } \tau) \)

\( (\downarrow_{\text{cbn}}) \ \frac{M_1 \downarrow_{\tau \rightarrow \tau'} \ \text{fn } x : \tau . M'_1 \quad M'_1[M_2/x] \downarrow_{\tau'} V}{M_1 \ M_2 \downarrow_{\tau'} V} \)
PCF evaluation (sample rules)

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

\[
(\downarrow_{\text{cbn}})
\begin{align*}
  M_1 & \downarrow_{\tau \rightarrow \tau'} \text{ fn } x : \tau . M'_1 & M'_1[M_2/x] & \downarrow_{\tau'} V \\
  M_1 & M_2 & \downarrow_{\tau'} V
\end{align*}
\]

\[
(\downarrow_{\text{fix}})
\begin{align*}
  M & \text{ fix}(M) & \downarrow_{\tau} V \\
  \text{ fix}(M) & \downarrow_{\tau} V
\end{align*}
\]
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 $M_1$, $M_2$, 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 denotational semantics — aims

- PCF types $\tau \mapsto$ domains $\llbracket \tau \rrbracket$. 
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.
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: $[M] = [M'] \Rightarrow [C[M]] = [C[M']]$. 
PCF denotational semantics — aims

- PCF types $\tau \mapsto$ domains $\llbracket \tau \rrbracket$.

- Closed PCF terms $M : \tau \mapsto$ elements $\llbracket M \rrbracket \in \llbracket \tau \rrbracket$.
  Denotations of open terms will be continuous functions.

- Compositionality.
  In particular: $\llbracket M \rrbracket = \llbracket M' \rrbracket \Rightarrow \llbracket C[M] \rrbracket = \llbracket C[M'] \rrbracket$.

- Soundness.
  For any type $\tau$, $M \Downarrow_\tau V \Rightarrow \llbracket M \rrbracket = \llbracket V \rrbracket$. 
PCF denotational semantics — aims

- PCF types $\tau \mapsto$ domains $\llbracket\tau\rrbracket$.

- Closed PCF terms $M : \tau \mapsto$ elements $\llbracket M \rrbracket \in \llbracket\tau\rrbracket$. Denotations of open terms will be continuous functions.

- Compositionality.
  In particular: $\llbracket M \rrbracket = \llbracket M' \rrbracket \Rightarrow \llbracket C[M] \rrbracket = \llbracket C[M'] \rrbracket$.

- Soundness.
  For any type $\tau$, $M \Downarrow_\tau V \Rightarrow \llbracket M \rrbracket = \llbracket V \rrbracket$.

- Adequacy.
  For $\tau = bool$ or $nat$, $\llbracket M \rrbracket = \llbracket V \rrbracket \in \llbracket\tau\rrbracket \Rightarrow M \Downarrow_\tau V$. 
Theorem. For all types $\tau$ and closed terms $M_1, M_2 \in \text{PCF}_\tau$, if $[M_1]$ and $[M_2]$ are equal elements of the domain $[\tau]$, then $M_1 \simeq_{\text{ctx}} M_2 : \tau$. 
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 $M_1 \simeq_{\text{ctx}} M_2 : \tau$.

Proof.

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

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

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

and symmetrically. \qed
Proof principle

To prove

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

it suffices to establish

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