Logical operations:

- If $S, T \subseteq D$ are chain-closed subsets of $D$ then
  
  $$S \cup T \quad \text{and} \quad S \cap T$$

  are chain-closed subsets of $D$.

- If $\{S_i\}_{i \in I}$ is a family of chain-closed subsets of $D$ indexed by a set $I$, then $\bigcap_{i \in I} S_i$ is a chain-closed subset of $D$.

- If a property $P(x, y)$ determines a chain-closed subset of $D \times E$, then the property $\forall x \in D. P(x, y)$ determines a chain-closed subset of $E$.
$S, T$ chain-closed $\Rightarrow$ $S U T$ chain-closed.

Consider $d_0 \leq d_1 \leq \ldots \leq d_n \leq \ldots$ in $S U T$.  

(1) $\{d_n\} \cap S$ finite.

Then there is an $N \in \mathbb{N}$ such that

$\forall n \geq N \in \mathbb{N}$ such that

$d_n \leq d_{n+1} \leq \ldots \leq d_{n+k} \leq \ldots$ in $T$

and

$\bigcup_n d_n = \lim_{n \to \infty} d_{n+k} \in T \subseteq S U T$

(2) $\{d_n\} \cap T$ finite

Analogous.
(3) \( \exists n \cup S \) and \( \{ d_n \cap T \} \in \infty_T \)

\[ \forall \left( \bigcup_{n} d_n \right) \in S \cup T \]

\[ \bigcup_{n} \{ d_n \cap S \} \in S \]

\[ \bigcup_{n} \{ d_n \cap T \} \in T \]
Lemma: \( \{ d_n \in \mathbb{N} \mid \exists e_n \in \mathbb{N}, \text{ in D} \} \)

Such that:

- for every \( d_n \) there exists an \( e_m \) such that \( d_n = e_m \)
- for every \( e_m \) there exists a \( d_n \) such that \( e_m \geq d_n \).

Then \( \bigcup_n d_n = \bigcup_m e_m \).
Example (III): Partial correctness

Let $\mathcal{F} : State \rightarrow State$ be the denotation of

$$\textbf{while } X > 0 \textbf{ do } (Y := X \ast Y; X := X - 1) .$$

For all $x, y \geq 0$,

$$\mathcal{F}[X \leftarrow x, Y \leftarrow y] \Downarrow \quad \implies \quad \mathcal{F}[X \leftarrow x, Y \leftarrow y] = [X \leftarrow 0, Y \leftarrow x! \cdot y].$$
Recall that

$$\mathcal{F} = \text{fix}(f)$$

where $f : (\text{State} \rightarrow \text{State}) \rightarrow (\text{State} \rightarrow \text{State})$ is given by

$$f(w) = \lambda (x, y) \in \text{State}. \begin{cases} (x, y) & \text{if } x \leq 0 \\ w(x - 1, x \cdot y) & \text{if } x > 0 \end{cases}$$
Proof by Scott induction.

We consider the admissible subset of \((\text{State} \rightarrow \text{State})\) given by

\[
S = \left\{ w \left| \begin{array}{l}
\forall x, y \geq 0. \\
w[X \mapsto x, Y \mapsto y] \downarrow \\
\Rightarrow w[X \mapsto x, Y \mapsto y] = [X \mapsto 0, Y \mapsto x! \cdot y]
\end{array} \right. \right\}
\]

and show that

\[
w \in S \implies f(w) \in S.
\]
Topic 5

PCF
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 \( x \in \mathbb{V} \), 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.

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

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

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

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

- Primitive recursion.

\[
H = \lambda x. (\lambda h. \lambda n. \text{if } (\text{zero } n) \text{ then } F x \text{ else } G x (\text{pred } n) (h x (\text{pred } n)))
\]

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

\[
F : \text{nat} \to \text{nat} \\
G : \text{nat} \to \text{nat} \to \text{nat} \to \text{nat}.
\]

\[
H : \text{nat} \to \text{nat} \to \text{nat}.
\]

\[
h x n = \text{if } (\text{zero } n) \text{ then } F x \text{ else } G x (\text{pred } n) (h x (\text{pred } n))
\]
Partial recursive functions in PCF

- Primitive recursion.

\[
F \ x \ n = \begin{cases} 
  \text{if } (\text{zero} (K \ x \ n)) \ \text{then } n \\
  \text{else } F \ x \ (\text{succ} \ n)
\end{cases}
\]

- Minimisation.

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

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

\[
K : \text{nat} \rightarrow \text{nat} \rightarrow \text{nat}
\]
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)

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

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

\[
\begin{align*}
(left) & \quad V \Downarrow_{\tau} V \quad (V \text{ a value of type } \tau) \\
\text{\(\Downarrow_{\text{val}}\)} & \\
\text{\(\Downarrow_{\text{cbn}}\)} & \\
& \quad M_1 \Downarrow_{\tau \rightarrow \tau'} \text{ fn } x : \tau \cdot M'_1 \quad M'_1[M_2/x] \Downarrow_{\tau'} V \\
& \quad M_1 M_2 \Downarrow_{\tau'} V \\
\text{\(\Downarrow_{\text{fix}}\)} & \\
& \quad M \text{ fix}(M) \Downarrow_{\tau} V \\
& \quad \text{ fix}(M) \Downarrow_{\tau} V
\end{align*}
\]
\[ \Omega = \text{fix}(\text{fn } x : \mathbb{Z}.x) : \mathbb{Z} \]

A non-terminating program.

\[ \text{fix}(\text{fn } x.x) \]

\[ \text{fix}(\text{fn } x.x) \]

\[ \text{fix}(\text{fn } x.x) \]

\[ (\text{fn } x.x)(\text{fix}(\text{fn } x.x)) \downarrow \]

---

\[ \text{fix}(\text{fn } x.x) = \Omega \downarrow \]
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 $V : \gamma$,

$$C[M_1] \Downarrow_{\gamma} V \iff C[M_2] \Downarrow_{\gamma} V.$$