Denotational Semantics

10 lectures for Part II CST 2016/17

Marcelo Fiore

Course web page:
http://www.cl.cam.ac.uk/teaching/1617/DenotSem/
Topic 1

Introduction
What is this course about?

• General area.

  *Formal methods*: Mathematical techniques for the specification, development, and verification of software and hardware systems.

• Specific area.

  *Formal semantics*: Mathematical theories for ascribing meanings to computer languages.
Why do we care?
Why do we care?

• Rigour.

  ... specification of programming languages
  ... justification of program transformations
Why do we care?

- Rigour.
  - specification of programming languages
  - justification of program transformations

- Insight.
  - generalisations of notions computability
  - higher-order functions
  - data structures
• Feedback into language design.
  
  ... continuations
  
  ... monads
● Feedback into language design.
  ... continuations
  ... monads

● Reasoning principles.
  ... Scott induction
  ... Logical relations
  ... Co-induction
Styles of formal semantics

Operational.

Axiomatic.

Denotational.
Styles of formal semantics

Operational.
Meanings for program phrases defined in terms of the *steps of computation* they can take during program execution.

Axiomatic.

Denotational.
Styles of formal semantics

Operational.
Meanings for program phrases defined in terms of the *steps of computation* they can take during program execution.

Axiomatic.
Meanings for program phrases defined indirectly via the *axioms and rules* of some logic of program properties.

Denotational.
Styles of formal semantics

Operational.
Meanings for program phrases defined in terms of the *steps of computation* they can take during program execution.

Axiomatic.
Meanings for program phrases defined indirectly via the *axioms and rules* of some logic of program properties.

Denotational.
Concerned with giving *mathematical models* of programming languages. Meanings for program phrases defined abstractly as elements of some suitable mathematical structure.
Basic idea of denotational semantics

Syntax $[-]$ \rightarrow Semantics

$P$ $\mapsto$ $[P]$
Basic idea of denotational semantics

Syntax $\rightarrow$ Semantics

Recursive program $\leftrightarrow$ Partial recursive function

$P \leftrightarrow \llbracket P \rrbracket$
Basic idea of denotational semantics

Syntax $\xrightarrow{[\cdot]}$ Semantics

Recursive program $\mapsto$ Partial recursive function

Boolean circuit $\mapsto$ Boolean function

$P$ $\mapsto$ $[P]$
Basic idea of denotational semantics

Syntax $\rightarrow$ Semantics

Recursive program $\mapsto$ Partial recursive function
Boolean circuit $\mapsto$ Boolean function

Concerns:

- Abstract models (i.e. implementation/machine independent).
  $\leadsto$ Lectures 2, 3 and 4.
Basic idea of denotational semantics

Syntax $\rightarrow$ Semantics

Recursive program $\mapsto$ Partial recursive function

Boolean circuit $\mapsto$ Boolean function

$P$ $\mapsto$ $[P]$

Concerns:

• Abstract models (i.e. implementation/machine independent).
  $\leadsto$ Lectures 2, 3 and 4.

• Compositionality.
  $\leadsto$ Lectures 5 and 6.
Basic idea of denotational semantics

Syntax $\rightarrow$ Semantics

Recursive program $\mapsto$ Partial recursive function

Boolean circuit $\mapsto$ Boolean function

$P \mapsto \llbracket P \rrbracket$

Concerns:

- Abstract models (i.e. implementation/machine independent).
  $\mapsto$ Lectures 2, 3 and 4.

- Compositionality.
  $\mapsto$ Lectures 5 and 6.

- Relationship to computation (e.g. operational semantics).
  $\mapsto$ Lectures 7 and 8.
Characteristic features of a denotational semantics

- Each phrase (= part of a program), \( P \), is given a denotation, \([P]\) — a mathematical object representing the contribution of \( P \) to the meaning of any complete program in which it occurs.

- The denotation of a phrase is determined just by the denotations of its subphrases (one says that the semantics is compositional).
Basic example of denotational semantics (I)

IMP− syntax

Arithmetic expressions

\[ A \in A_{\text{exp}} ::= n \mid L \mid A + A \mid \ldots \]
where \( n \) ranges over integers and \( L \) over a specified set of locations \( \mathbb{L} \)

Boolean expressions

\[ B \in B_{\text{exp}} ::= \text{true} \mid \text{false} \mid A = A \mid \ldots \]
\[ \mid \neg B \mid \ldots \]

Commands

\[ C \in \text{Comm} ::= \text{skip} \mid L := A \mid C; C \]
\[ \mid \text{if } B \text{ then } C \text{ else } C \]
Basic example of denotational semantics (II)

Semantic functions

\[ A : \text{Aexp} \rightarrow (\text{State} \rightarrow \mathbb{Z}) \]

where

\[ \mathbb{Z} = \{ \ldots, -1, 0, 1, \ldots \} \]

\[ \text{State} = (\mathbb{L} \rightarrow \mathbb{Z}) \]
Basic example of denotational semantics (II)

Semantic functions

\[ \mathcal{A} : \text{Aexp} \rightarrow (\text{State} \rightarrow \mathbb{Z}) \]
\[ \mathcal{B} : \text{Bexp} \rightarrow (\text{State} \rightarrow \mathbb{B}) \]

where

\[ \mathbb{Z} = \{ \ldots, -1, 0, 1, \ldots \} \]
\[ \mathbb{B} = \{ \text{true}, \text{false} \} \]
\[ \text{State} = (\mathcal{L} \rightarrow \mathbb{Z}) \]
Basic example of denotational semantics (II)

Semantic functions

\[ A : \text{Aexp} \rightarrow (\text{State} \rightarrow \mathbb{Z}) \]
\[ B : \text{Bexp} \rightarrow (\text{State} \rightarrow \mathbb{B}) \]
\[ C : \text{Comm} \rightarrow (\text{State} \rightarrow \text{State}) \]

where

\[ \mathbb{Z} = \{ \ldots, -1, 0, 1, \ldots \} \]
\[ \mathbb{B} = \{ \text{true}, \text{false} \} \]
\[ \text{State} = (\mathbb{L} \rightarrow \mathbb{Z}) \]
Basic example of denotational semantics (III)

Semantic function $\mathcal{A}$

\[
\mathcal{A}[n] = \lambda s \in \text{State}. n \\
\mathcal{A}[L] = \lambda s \in \text{State}. s(L) \\
\mathcal{A}[A_1 + A_2] = \lambda s \in \text{State}. \mathcal{A}[A_1](s) + \mathcal{A}[A_2](s)
\]
Basic example of denotational semantics (IV)

Semantic function $\mathcal{B}$

\[
\begin{align*}
\mathcal{B}[\text{true}] &= \lambda s \in \text{State}. \text{true} \\
\mathcal{B}[\text{false}] &= \lambda s \in \text{State}. \text{false} \\
\mathcal{B}[A_1 = A_2] &= \lambda s \in \text{State}. \text{eq}(\mathcal{A}[A_1](s), \mathcal{A}[A_2](s))
\end{align*}
\]

where $\text{eq}(a, a') = \begin{cases} 
\text{true} & \text{if } a = a' \\
\text{false} & \text{if } a \neq a'
\end{cases}$
Basic example of denotational semantics (V)

Semantic function $\mathcal{C}$

$$\left[\text{skip}\right] = \lambda s \in \text{State}. \ s$$

NB: From now on the names of semantic functions are omitted!
A simple example of compositionality

Given partial functions $\left[ C \right], \left[ C' \right] : State \rightarrow State$ and a function $\left[ B \right] : State \rightarrow \{true, false\}$, we can define

$$\left[ \text{if } B \text{ then } C \text{ else } C' \right] = \lambda s \in State. \text{if} (\left[ B \right](s), \left[ C \right](s), \left[ C' \right](s))$$

where

$$\text{if } (b, x, x') = \begin{cases} x & \text{if } b = true \\ x' & \text{if } b = false \end{cases}$$
Basic example of denotational semantics (VI)

Semantic function $C$

$$
[L := A] = \lambda s \in \text{State}. \lambda \ell \in \mathbb{L}. \text{if } (\ell = L, [A](s), s(\ell))
$$
Denotational semantics of sequential composition

Denotation of sequential composition $C; C'$ of two commands

$$[C; C'] = [C'] \circ [C] = \lambda s \in State. [C']([C](s))$$

given by composition of the partial functions from states to states $[C], [C'] : State \rightarrow State$ which are the denotations of the commands.
Denotational semantics of sequential composition

Denotation of sequential composition $C; C'$ of two commands

$$\left[ C; C' \right] = \left[ C' \right] \circ \left[ C \right] = \lambda s \in \text{State}. \left[ C' \right]\left( \left[ C \right](s) \right)$$

given by composition of the partial functions from states to states $\left[ C \right], \left[ C' \right]: \text{State} \rightarrow \text{State}$ which are the denotations of the commands.

Cf. operational semantics of sequential composition:

$$C, s \Downarrow s' \quad C', s' \Downarrow s'' \quad \therefore \quad C; C', s \Downarrow s''$$
[while $B$ do $C$]
Fixed point property of

\[ [\text{while } B \text{ do } C] \]

\[ [\text{while } B \text{ do } C] = f_{[B],[C]}([\text{while } B \text{ do } C]) \]

where, for each \( b : \text{State} \rightarrow \{ \text{true, false} \} \) and \( c : \text{State} \rightarrow \text{State} \), we define

\[ f_{b,c} : (\text{State} \rightarrow \text{State}) \rightarrow (\text{State} \rightarrow \text{State}) \]

as

\[ f_{b,c} = \lambda w \in (\text{State} \rightarrow \text{State}). \lambda s \in \text{State}. \text{if} (b(s), w(c(s)), s). \]
Fixed point property of 
\[\texttt{[while } B \texttt{ do } C \texttt{]}\]

\[
\texttt{[while } B \texttt{ do } C \texttt{]} = f_{[B],[C]}(\texttt{[while } B \texttt{ do } C \texttt{]})
\]

where, for each \(b : \text{State} \rightarrow \{\text{true, false}\}\) and \(c : \text{State} \rightarrow \text{State}\), we define

\[
f_{b,c} : (\text{State} \rightarrow \text{State}) \rightarrow (\text{State} \rightarrow \text{State})
\]

as

\[
f_{b,c} = \lambda w \in (\text{State} \rightarrow \text{State}). \lambda s \in \text{State}. \text{if } (b(s), w(c(s)), s).
\]

- Why does \(w = f_{[B],[C]}(w)\) have a solution?
- What if it has several solutions—which one do we take to be \(\texttt{[while } B \texttt{ do } C \texttt{]}\)?
Approximating $[\text{while } B \text{ do } C]$
Approximating \([\text{while } B \text{ do } C]\)

\[f_{[B],[C]}^n(\bot) = \lambda s \in \text{State.} \]

\[
\begin{cases} 
[C]^k(s) & \text{if } \exists 0 \leq k < n. [B]([C]^k(s)) = false \\
& \text{and } \forall 0 \leq i < k. [B]([C]^i(s)) = true \\
\uparrow & \text{if } \forall 0 \leq i < n. [B]([C]^i(s)) = true
\end{cases}
\]
\[ D \overset{\text{def}}{=} (\text{State} \rightarrow \text{State}) \]

- **Partial order \( \sqsubseteq \) on \( D \):**
  
  \[ w \sqsubseteq w' \quad \text{iff} \quad \text{for all } s \in \text{State}, \text{ if } w \text{ is defined at } s \text{ then so is } w' \text{ and moreover } w(s) = w'(s). \]
  
  \[ \text{ iff the graph of } w \text{ is included in the graph of } w'. \]

- **Least element \( \bot \in D \) w.r.t. \( \sqsubseteq \):**
  
  \[ \bot = \text{totally undefined partial function} \]
  
  \[ = \text{partial function with empty graph} \]

  (satisfies \( \bot \sqsubseteq w \), for all \( w \in D \)).
Topic 2

Least Fixed Points
All domains of computation are partial orders with a least element.
Thesis

All domains of computation are partial orders with a least element.

All computable functions are monotonotic.
A binary relation \( \sqsubseteq \) on a set \( D \) is a partial order iff it is

**reflexive:** \( \forall d \in D. \ d \sqsubseteq d \)

**transitive:** \( \forall d, d', d'' \in D. \ d \sqsubseteq d' \sqsubseteq d'' \Rightarrow d \sqsubseteq d'' \)

**anti-symmetric:** \( \forall d, d' \in D. \ d \sqsubseteq d' \sqsubseteq d \Rightarrow d = d' \).

Such a pair \( (D, \sqsubseteq) \) is called a partially ordered set, or poset.
\[ x \subseteq x \]

\[ x \subseteq y \quad y \subseteq z \]

\[ x \subseteq z \]

\[ x \subseteq y \quad y \subseteq x \]

\[ x = y \]
Domain of partial functions, $X \twoheadrightarrow Y$
Domain of partial functions, \( X \rightarrow Y \)

**Underlying set:** all partial functions, \( f \), with domain of definition \( \text{dom}(f) \subseteq X \) and taking values in \( Y \).
Domain of partial functions, \( X \rightarrow Y \)

**Underlying set:** all partial functions, \( f \), with domain of definition \( \text{dom}(f) \subseteq X \) and taking values in \( Y \).

**Partial order:**

\[
f \sqsubseteq g \quad \text{iff} \quad \text{dom}(f) \subseteq \text{dom}(g) \quad \text{and} \quad \forall x \in \text{dom}(f). \ f(x) = g(x)
\]

iff \( \text{graph}(f) \subseteq \text{graph}(g) \)
Monotonicity

- A function $f : D \rightarrow E$ between posets is **monotone** iff
  \[ \forall d, d' \in D. \ d \sqsubseteq d' \Rightarrow f(d) \sqsubseteq f(d'). \]
Least Elements

Suppose that $D$ is a poset and that $S$ is a subset of $D$.

An element $d \in S$ is the least element of $S$ if it satisfies

$$\forall x \in S. \quad d \sqsubseteq x .$$

- Note that because $\sqsubseteq$ is anti-symmetric, $S$ has at most one least element.

- Note also that a poset may not have least element.
Pre-fixed points

Let $D$ be a poset and $f : D \to D$ be a function.

An element $d \in D$ is a pre-fixed point of $f$ if it satisfies $f(d) \sqsubseteq d$.

The least pre-fixed point of $f$, if it exists, will be written $\text{fix}(f)$.

It is thus (uniquely) specified by the two properties:

\begin{align*}
    f(\text{fix}(f)) & \sqsubseteq \text{fix}(f) \quad \text{(lfp1)} \\
    \forall d \in D. \; f(d) & \sqsubseteq d \Rightarrow \text{fix}(f) \sqsubseteq d. \quad \text{(lfp2)}
\end{align*}
2. Let $D$ be a poset and let $f : D \to D$ be a function with a least pre-fixed point $\text{fix}(f) \in D$.

For all $x \in D$, to prove that $\text{fix}(f) \sqsubseteq x$ it is enough to establish that $f(x) \sqsubseteq x$. 
Proof principle

2. Let $D$ be a poset and let $f : D \to D$ be a function with a least pre-fixed point $\text{fix}(f) \in D$.

For all $x \in D$, to prove that $\text{fix}(f) \sqsubseteq x$ it is enough to establish that $f(x) \sqsubseteq x$.

$$f(x) \sqsubseteq x$$

$$\underline{\text{fix}(f) \sqsubseteq x}$$
1. \[ f(\text{fix}(f)) \sqsubseteq \text{fix}(f) \]

2. Let \( D \) be a poset and let \( f : D \to D \) be a function with a least pre-fixed point \( \text{fix}(f) \in D \).
For all \( x \in D \), to prove that \( \text{fix}(f) \sqsubseteq x \) it is enough to establish that \( f(x) \sqsubseteq x \).

\[ f(x) \sqsubseteq x \]
\[ \implies \text{fix}(f) \sqsubseteq x \]
Least pre-fixed points are fixed points

If it exists, the least pre-fixed point of a mononote function on a partial order is necessarily a fixed point.
All domains of computation are complete partial orders with a least element.
All domains of computation are complete partial orders with a least element.

All computable functions are continuous.
Cpo’s and domains

A chain complete poset, or cpo for short, is a poset \((D, \sqsubseteq)\) in which all countable increasing chains
\[
d_0 \sqsubseteq d_1 \sqsubseteq d_2 \sqsubseteq \ldots
\]
have least upper bounds,
\[
\bigcup_{n \geq 0} d_n:
\]

\[
\forall m \geq 0 . \ d_m \sqsubseteq \bigcup_{n \geq 0} d_n \tag{ lub1} \]

\[
\forall d \in D . \ (\forall m \geq 0 . \ d_m \sqsubseteq d) \Rightarrow \bigcup_{n \geq 0} d_n \sqsubseteq d. \tag{ lub2}
\]

A domain is a cpo that possesses a least element, \(\bot\):

\[
\forall d \in D . \ \bot \sqsubseteq d.
\]
⊥ ⊑ x

\[ x_i \sqsubseteq \bigcup_{n \geq 0} x_n \quad (i \geq 0 \text{ and } \langle x_n \rangle \text{ a chain}) \]

\[ \forall n \geq 0 . x_n \sqsubseteq x \]

\[ \bigcup_{n \geq 0} x_n \subseteq x \quad (\langle x_i \rangle \text{ a chain}) \]
Domain of partial functions, $X \rightarrow Y$
Domain of partial functions, $X \rightarrow Y$

**Underlying set:** all partial functions, $f$, with domain of definition $\text{dom}(f) \subseteq X$ and taking values in $Y$. 
Domain of partial functions, $X \to Y$

**Underlying set:** all partial functions, $f$, with domain of definition $\text{dom}(f) \subseteq X$ and taking values in $Y$.

**Partial order:**

$f \sqsubseteq g$ \iff \begin{align*}
\text{dom}(f) & \subseteq \text{dom}(g) \text{ and } \\
\forall x \in \text{dom}(f). \ f(x) & = g(x)
\end{align*}

\iff \text{graph}(f) \subseteq \text{graph}(g)$
Domain of partial functions, $X \rightarrow Y$

**Underlying set:** all partial functions, $f$, with domain of definition $\text{dom}(f) \subseteq X$ and taking values in $Y$.

**Partial order:**

\[
 f \sqsubseteq g \iff \text{dom}(f) \subseteq \text{dom}(g) \text{ and } \forall x \in \text{dom}(f). f(x) = g(x)
\]

iff $\text{graph}(f) \subseteq \text{graph}(g)$

**Lub of chain** $f_0 \sqsubseteq f_1 \sqsubseteq f_2 \sqsubseteq \ldots$ is the partial function $f$ with $\text{dom}(f) = \bigcup_{n \geq 0} \text{dom}(f_n)$ and

\[
 f(x) = \begin{cases} 
 f_n(x) & \text{if } x \in \text{dom}(f_n), \text{ some } n \\
 \text{undefined} & \text{otherwise}
\end{cases}
\]
Domain of partial functions, $X \rightarrow Y$

**Underlying set:** all partial functions, $f$, with domain of definition $\text{dom}(f) \subseteq X$ and taking values in $Y$.

**Partial order:**

$f \sqsubseteq g$ if and only if $\text{dom}(f) \subseteq \text{dom}(g)$ and $\forall x \in \text{dom}(f). \ f(x) = g(x)$

iff $\text{graph}(f) \subseteq \text{graph}(g)$

**Lub of chain** $f_0 \sqsubseteq f_1 \sqsubseteq f_2 \sqsubseteq \ldots$ is the partial function $f$ with $\text{dom}(f) = \bigcup_{n \geq 0} \text{dom}(f_n)$ and

$$f(x) = \begin{cases} f_n(x) & \text{if } x \in \text{dom}(f_n), \text{ some } n \\ \text{undefined} & \text{otherwise} \end{cases}$$

**Least element** $\bot$ is the totally undefined partial function.
Some properties of lubs of chains

Let $D$ be a cpo.

1. For $d \in D$, $\bigsqcup_n d = d$.

2. For every chain $d_0 \sqsubseteq d_1 \sqsubseteq \ldots \sqsubseteq d_n \sqsubseteq \ldots$ in $D$,

$$\bigsqcup_n d_n = \bigsqcup_n d_{N+n}$$

for all $N \in \mathbb{N}$. 
3. For every pair of chains \( d_0 \sqsubseteq d_1 \sqsubseteq \ldots \sqsubseteq d_n \sqsubseteq \ldots \) and \( e_0 \sqsubseteq e_1 \sqsubseteq \ldots \sqsubseteq e_n \sqsubseteq \ldots \) in \( D \), if \( d_n \sqsubseteq e_n \) for all \( n \in \mathbb{N} \) then \( \bigcup_n d_n \sqsubseteq \bigcup_n e_n \).
3. For every pair of chains $d_0 \sqsubseteq d_1 \sqsubseteq \ldots \sqsubseteq d_n \sqsubseteq \ldots$ and $e_0 \sqsubseteq e_1 \sqsubseteq \ldots \sqsubseteq e_n \sqsubseteq \ldots$ in $D$, if $d_n \sqsubseteq e_n$ for all $n \in \mathbb{N}$ then $\bigcup_n d_n \sqsubseteq \bigcup_n e_n$.

$$\forall n \geq 0 . \ x_n \sqsubseteq y_n \quad \Rightarrow \quad \bigcup_n x_n \sqsubseteq \bigcup_n y_n$$

(\langle x_n \rangle \text{ and } \langle y_n \rangle \text{ chains})
Lemma. Let $D$ be a cpo. Suppose that the doubly-indexed family of elements $d_{m,n} \in D$ ($m, n \geq 0$) satisfies

$$m \leq m' \& n \leq n' \Rightarrow d_{m,n} \sqsubseteq d_{m',n'}.$$  

Then

$$\bigsqcup_{n \geq 0} d_{0,n} \sqsubseteq \bigsqcup_{n \geq 0} d_{1,n} \sqsubseteq \bigsqcup_{n \geq 0} d_{2,n} \sqsubseteq \ldots$$

and

$$\bigsqcup_{m \geq 0} d_{m,0} \sqsubseteq \bigsqcup_{m \geq 0} d_{m,1} \sqsubseteq \bigsqcup_{m \geq 0} d_{m,3} \sqsubseteq \ldots.$$
Diagonalising a double chain

**Lemma.** Let $D$ be a cpo. Suppose that the doubly-indexed family of elements $d_{m,n} \in D \ (m, n \geq 0)$ satisfies

$$m \leq m' \ \& \ n \leq n' \ \Rightarrow \ d_{m,n} \sqsubseteq d_{m',n'}.$$  

(†)

Then

$$\bigsqcup_{n\geq 0} d_{0,n} \sqsubseteq \bigsqcup_{n\geq 0} d_{1,n} \sqsubseteq \bigsqcup_{n\geq 0} d_{2,n} \sqsubseteq \ldots$$

and

$$\bigsqcup_{m\geq 0} d_{m,0} \sqsubseteq \bigsqcup_{m\geq 0} d_{m,1} \sqsubseteq \bigsqcup_{m\geq 0} d_{m,3} \sqsubseteq \ldots$$

Moreover

$$\bigsqcup_{m\geq 0} \left( \bigsqcup_{n\geq 0} d_{m,n} \right) = \bigsqcup_{k\geq 0} d_{k,k} = \bigsqcup_{n\geq 0} \left( \bigsqcup_{m\geq 0} d_{m,n} \right).$$
Continuity and strictness

- If $D$ and $E$ are cpo’s, the function $f$ is continuous iff
  1. it is monotone, and
  2. it preserves lubs of chains, i.e. for all chains $d_0 \sqsubseteq d_1 \sqsubseteq \ldots$ in $D$, it is the case that

$$f\left(\bigsqcup_{n \geq 0} d_n\right) = \bigsqcup_{n \geq 0} f(d_n) \quad \text{in } E.$$
Continuity and strictness

• If $D$ and $E$ are cpo’s, the function $f$ is continuous iff
  1. it is monotone, and
  2. it preserves lubs of chains, i.e. for all chains $d_0 \sqsubseteq d_1 \sqsubseteq \ldots$ in $D$, it is the case that
     \[
     f(\bigsqcup_{n \geq 0} d_n) = \bigsqcup_{n \geq 0} f(d_n) \quad \text{in } E.
     \]

• If $D$ and $E$ have least elements, then the function $f$ is strict iff $f(\bot) = \bot$. 

Tarski’s Fixed Point Theorem

Let \( f : D \to D \) be a continuous function on a domain \( D \). Then

- \( f \) possesses a least pre-fixed point, given by
  \[
  \text{fix}(f) = \bigsqcup_{n \geq 0} f^n(\bot).
  \]

- Moreover, \( \text{fix}(f) \) is a fixed point of \( f \), i.e. satisfies
  \[
  f(\text{fix}(f)) = \text{fix}(f),
  \]
  and hence is the least fixed point of \( f \).
\[ \text{[while } B \text{ do } C] \]

\[ = \text{fix}(f_{[B],[C]}) \]

\[ = \bigcup_{n \geq 0} f_{[B],[C]}^n(\bot) \]

\[ = \lambda s \in \text{State.} \]

\[
\begin{cases} 
[C]^k(s) & \text{if } k \geq 0 \text{ is such that } [B]([C]^k(s)) = \text{false} \\
\text{undefined} & \text{if } [B]([C]^i(s)) = \text{true} \text{ for all } i \geq 0 \\
\text{true} \text{ for all } 0 \leq i < k 
\end{cases}
\]
Topic 3

Constructions on Domains
Discrete cpo’s and flat domains

For any set $X$, the relation of equality

$$x \sqsubseteq x' \overset{\text{def}}{\iff} x = x' \quad (x, x' \in X)$$

makes $(X, \sqsubseteq)$ into a cpo, called the discrete cpo with underlying set $X$. 
Discrete cpo’s and flat domains

For any set $X$, the relation of equality

\[ x \sqsubseteq x' \iff x = x' \quad (x, x' \in X) \]

makes $(X, \sqsubseteq)$ into a cpo, called the discrete cpo with underlying set $X$.

Let $X_\bot \overset{\text{def}}{=} X \cup \{\bot\}$, where $\bot$ is some element not in $X$. Then

\[ d \sqsubseteq d' \overset{\text{def}}{=} (d = d') \lor (d = \bot) \quad (d, d' \in X_\bot) \]

makes $(X_\bot, \sqsubseteq)$ into a domain (with least element $\bot$), called the flat domain determined by $X$. 

Binary product of cpo’s and domains

The **product** of two cpo’s \((D_1, \sqsubseteq_1)\) and \((D_2, \sqsubseteq_2)\) has underlying set

\[
D_1 \times D_2 = \{(d_1, d_2) \mid d_1 \in D_1 \& d_2 \in D_2\}
\]

and partial order \(\sqsubseteq\) defined by

\[
(d_1, d_2) \sqsubseteq (d'_1, d'_2) \overset{\text{def}}{\iff} d_1 \sqsubseteq_1 d'_1 \& d_2 \sqsubseteq_2 d'_2.
\]
Lubs of chains are calculated componentwise:

$$\bigsqcup_{n \geq 0} (d_{1,n}, d_{2,n}) = (\bigsqcup_{i \geq 0} d_{1,i}, \bigsqcup_{j \geq 0} d_{2,j}) .$$

If $\mathcal{(D_1, \sqsubseteq_1)}$ and $\mathcal{(D_2, \sqsubseteq_2)}$ are domains so is $\mathcal{(D_1 \times D_2, \sqsubseteq)}$ and $\bot_{D_1 \times D_2} = (\bot_{D_1}, \bot_{D_2})$. 
Continuous functions of two arguments

Proposition. Let \( D, E, F \) be cpo’s. A function \( f : (D \times E) \to F \) is monotone if and only if it is monotone in each argument separately:

\[
\forall d, d' \in D, e \in E. \quad d \sqsubseteq d' \Rightarrow f(d, e) \sqsubseteq f(d', e)
\]

\[
\forall d \in D, e, e' \in E. \quad e \sqsubseteq e' \Rightarrow f(d, e) \sqsubseteq f(d, e').
\]

Moreover, it is continuous if and only if it preserves lubs of chains in each argument separately:

\[
f(\bigsqcup_{m \geq 0} d_m, e) = \bigsqcup_{m \geq 0} f(d_m, e)
\]

\[
f(d, \bigsqcup_{n \geq 0} e_n) = \bigsqcup_{n \geq 0} f(d, e_n).
\]
• A couple of derived rules:

\[
\begin{array}{c}
x \sqsubseteq x' \\
y \sqsubseteq y'
\end{array}
\Rightarrow
\frac{f(x, y) \sqsubseteq f(x', y')}{(f \text{ monotone})}
\]

\[
f(\bigsqcup_m x_m, \bigsqcup_n y_n) = \bigsqcup_k f(x_k, y_k)
\]
Given cpo’s \((D, \sqsubseteq_D)\) and \((E, \sqsubseteq_E)\), the function cpo \((D \rightarrow E, \sqsubseteq)\) has underlying set

\[
(D \rightarrow E) \overset{\text{def}}{=} \{ f \mid f : D \rightarrow E \text{ is a continuous function} \}
\]

and partial order: \(f \sqsubseteq f' \overset{\text{def}}{=} \forall d \in D . f(d) \sqsubseteq_E f'(d)\).
Function cpo’s and domains

Given cpo’s \((D, \sqsubseteq_D)\) and \((E, \sqsubseteq_E)\), the function cpo \((D \to E, \sqsubseteq)\) has underlying set

\[
(D \to E) \overset{\text{def}}{=} \{ f \mid f : D \to E \text{ is a continuous function} \}
\]

and partial order: \(f \sqsubseteq f' \overset{\text{def}}{\iff} \forall d \in D . f(d) \sqsubseteq_E f'(d).\)

- A derived rule:

\[
\frac{f \sqsubseteq (D \to E) \quad g \quad x \sqsubseteq_D y}{f(x) \sqsubseteq g(y)}
\]
Lubs of chains are calculated ‘argumentwise’ (using lubs in \( E \)):

\[
\bigsqcup_{n \geq 0} f_n = \lambda d \in D. \, \bigsqcup_{n \geq 0} f_n(d).
\]

If \( E \) is a domain, then so is \( D \rightarrow E \) and \( \bot_{D \rightarrow E}(d) = \bot_E \), all \( d \in D \).
Lubs of chains are calculated ‘argumentwise’ (using lubs in $E$): 

$$\bigsqcup_{n \geq 0} f_n = \lambda d \in D. \bigsqcup_{n \geq 0} f_n(d).$$

- A derived rule:

$$\left( \bigsqcup_n f_n \right) \left( \bigsqcup_m x_m \right) = \bigsqcup_k f_k(x_k)$$

If $E$ is a domain, then so is $D \to E$ and $\bot_{D \to E}(d) = \bot_E$, all $d \in D$. 

47
For cpo’s $D, E, F$, the composition function

$$\circ : \left( (E \to F) \times (D \to E) \right) \longrightarrow (D \to F)$$

defined by setting, for all $f \in (D \to E)$ and $g \in (E \to F)$,

$$g \circ f = \lambda d \in D. \ g(f(d))$$

is continuous.
Continuity of the fixpoint operator

Let $D$ be a domain.

By Tarski’s Fixed Point Theorem we know that each continuous function $f \in (D \rightarrow D)$ possesses a least fixed point, $\text{fix}(f) \in D$.

**Proposition.** The function

$$\text{fix} : (D \rightarrow D) \rightarrow D$$

is continuous.
Topic 4

Scott Induction
Scott’s Fixed Point Induction Principle

Let \( f : D \rightarrow D \) be a continuous function on a domain \( D \).

For any admissible subset \( S \subseteq D \), to prove that the least fixed point of \( f \) is in \( S \), i.e. that

\[
\text{fix}(f) \in S,
\]

it suffices to prove

\[
\forall d \in D \ (d \in S \Rightarrow f(d) \in S).
\]
Chain-closed and admissible subsets

Let $D$ be a cpo. A subset $S \subseteq D$ is called chain-closed iff for all chains $d_0 \sqsubseteq d_1 \sqsubseteq d_2 \sqsubseteq \ldots$ in $D$

$$(\forall n \geq 0. \ d_n \in S) \Rightarrow \left( \bigsqcup_{n\geq 0} d_n \right) \in S$$

If $D$ is a domain, $S \subseteq D$ is called admissible iff it is a chain-closed subset of $D$ and $\perp \in S$. 
Chain-closed and admissible subsets

Let $D$ be a cpo. A subset $S \subseteq D$ is called chain-closed iff for all chains $d_0 \sqsubseteq d_1 \sqsubseteq d_2 \sqsubseteq \ldots$ in $D$

$$(\forall n \geq 0 . d_n \in S) \Rightarrow \left( \bigsqcup_{n \geq 0} d_n \right) \in S$$

If $D$ is a domain, $S \subseteq D$ is called admissible iff it is a chain-closed subset of $D$ and $\perp \in S$.

A property $\Phi(d)$ of elements $d \in D$ is called chain-closed (resp. admissible) iff $\{d \in D \mid \Phi(d)\}$ is a chain-closed (resp. admissible) subset of $D$. 
Building chain-closed subsets (I)

Let $D, E$ be cpos.

**Basic relations:**

- For every $d \in D$, the subset

$$\downarrow(d) \overset{\text{def}}{=} \{ x \in D \mid x \sqsubseteq d \}$$

of $D$ is chain-closed.
Let $D$, $E$ be cpos.

**Basic relations:**

- For every $d \in D$, the subset
  \[ \downarrow(d) \overset{\text{def}}{=} \{ x \in D \mid x \sqsubseteq d \} \]
  of $D$ is chain-closed.

- The subsets
  \[ \{ (x, y) \in D \times D \mid x \sqsubseteq y \} \]
  and
  \[ \{ (x, y) \in D \times D \mid x = y \} \]
  of $D \times D$ are chain-closed.
Example (I): Least pre-fixed point property

Let $D$ be a domain and let $f : D \rightarrow D$ be a continuous function.

$$\forall d \in D. f(d) \sqsubseteq d \implies \text{fix}(f) \sqsubseteq d$$
Example (I): Least pre-fixed point property

Let $D$ be a domain and let $f : D \to D$ be a continuous function.

\[ \forall d \in D. \ f(d) \sqsubseteq d \implies \text{fix}(f) \sqsubseteq d \]

**Proof by Scott induction.**

Let $d \in D$ be a pre-fixed point of $f$. Then,

\[ x \in \downarrow(d) \implies x \sqsubseteq d \]
\[ \implies f(x) \sqsubseteq f(d) \]
\[ \implies f(x) \sqsubseteq d \]
\[ \implies f(x) \in \downarrow(d) \]

Hence,

\[ \text{fix}(f) \in \downarrow(d) \]
Inverse image:

Let \( f : D \to E \) be a continuous function.

If \( S \) is a chain-closed subset of \( E \) then the inverse image

\[
f^{-1}S = \{ x \in D \mid f(x) \in S \}
\]

is an chain-closed subset of \( D \).
Example (II)

Let $D$ be a domain and let $f, g : D \to D$ be continuous functions such that $f \circ g \sqsubseteq g \circ f$. Then,

$$f(\bot) \sqsubseteq g(\bot) \implies \operatorname{fix}(f) \sqsubseteq \operatorname{fix}(g).$$
Example (II)

Let $D$ be a domain and let $f, g : D \to D$ be continuous functions such that $f \circ g \sqsubseteq g \circ f$. Then,

$$f(\bot) \sqsubseteq g(\bot) \implies \text{fix}(f) \sqsubseteq \text{fix}(g).$$

**Proof by Scott induction.**

Consider the admissible property $\Phi(x) \equiv (f(x) \sqsubseteq g(x))$ of $D$.

Since

$$f(x) \sqsubseteq g(x) \implies g(f(x)) \sqsubseteq g(g(x)) \implies f(g(x)) \sqsubseteq g(g(x))$$

we have that

$$f(\text{fix}(g)) \sqsubseteq g(\text{fix}(g)).$$
Logical operations:

- If $S, T \subseteq D$ are chain-closed subsets of $D$ then $S \cup T$ and $S \cap T$ 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 $E$. 

Building chain-closed subsets (III)
Example (III): Partial correctness

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

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

For all $x, y \geq 0$,

$$\mathcal{F}[X \mapsto x, Y \mapsto y] \downarrow$$

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

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

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

\[
S = \left\{ w \mid \forall x, y \geq 0. \quad w[X \mapsto x, Y \mapsto y] \downarrow \Rightarrow w[X \mapsto x, Y \mapsto y] = [X \mapsto 0, Y \mapsto !x \cdot y] \right\}
\]

and show that

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

PCF
PCF syntax

Types

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

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

Expressions

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

Types

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

Expressions

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

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

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

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

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 \textit{variables}.

\textbf{Technicality:} We identify expressions up to \( \alpha \)-conversion of bound variables (created by the \text{fn} expression-former): by definition a PCF \textit{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)

\[\begin{array}{c}
\Gamma[x \mapsto \tau] \vdash M : \tau' \\
\Gamma \vdash \text{fn } x : \tau . M : \tau \rightarrow \tau' \\
\hline
\end{array}\]

if \( x \notin \text{dom}(\Gamma) \)
PCF typing relation (sample rules)

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

\[
\begin{align*}
(\text{:fn}) & \quad \frac{\Gamma[x \mapsto \tau] \vdash M : \tau'}{\Gamma \vdash \text{fn} \, x : \tau \cdot M : \tau \to \tau'} \quad \text{if } x \notin \text{dom}(\Gamma) \\
(\text{:app}) & \quad \frac{\Gamma \vdash M_1 : \tau \to \tau' \quad \Gamma \vdash M_2 : \tau}{\Gamma \vdash M_1 \, M_2 : \tau'} \\
(\text{:fix}) & \quad \frac{\Gamma \vdash M : \tau \to \tau}{\Gamma \vdash \text{fix}(M) : \tau}
\end{align*}
\]
Partial recursive functions in PCF

• Primitive recursion.

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

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

\[
\begin{align*}
(\downarrow_{\text{val}}) & \quad V \downarrow_{\tau} V \quad (V \text{ a value of type } \tau) \\
(\downarrow_{\text{cbn}}) & \quad \frac{M_1 \downarrow_{\tau \rightarrow \tau'} \ \textbf{fn} \ x : \tau \ . \ M'_1 \quad M'_1[M_2/x] \downarrow_{\tau'} V}{M_1 \ M_2 \downarrow_{\tau'} V} \\
(\downarrow_{\text{fix}}) & \quad \frac{M \ \textbf{fix}(M) \downarrow_{\tau} V}{\textbf{fix}(M) \downarrow_{\tau} V}
\end{align*}
\]
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.
Contextual equivalence of PCF terms

Given PCF terms $M_1$, $M_2$, PCF type $\tau$, and a type environment $\Gamma$, the relation $\Gamma \vdash M_1 \cong_{\text{ctx}} M_2 : \tau$ 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 $\tau \mapsto \text{domains } [\tau]$.

- Closed PCF terms $M : \tau \mapsto \text{elements } [M] \in [\tau]$.
  Denotations of open terms will be continuous functions.

- Compositionality.
  In particular: $[M] = [M'] \Rightarrow [C[M]] = [C[M']]$. 
PCF denotational semantics — aims

- PCF types $\tau \mapsto \text{domains } [\tau]$.

- Closed PCF terms $M : \tau \mapsto \text{elements } [M] \in [\tau]$. Denotations of open terms will be continuous functions.

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

- Soundness.
  For any type $\tau$, $M \Downarrow_\tau V \Rightarrow [M] = [V]$. 
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 $[M_1]$ 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.

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

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

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

and symmetrically. \hfill \Box
Proof principle

To prove

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

it suffices to establish

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

To prove

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

it suffices to establish

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

The proof principle is sound, but is it complete? That is, is equality in the denotational model also a necessary condition for contextual equivalence?
Topic 6

Denotational Semantics of PCF
Denotational semantics of PCF

To every typing judgement

\[ \Gamma \vdash M : \tau \]

we associate a continuous function

\[ [\Gamma \vdash M] : [\Gamma] \rightarrow [\tau] \]

between domains.
Denotational semantics of PCF types

\[
\lbrack nat \rbrack \overset{\text{def}}{=} N_\perp \quad \text{(flat domain)}
\]

\[
\lbrack bool \rbrack \overset{\text{def}}{=} B_\perp \quad \text{(flat domain)}
\]

where \( N = \{0, 1, 2, \ldots\} \) and \( B = \{true, false\} \).
Denotational semantics of PCF types

\[
\begin{align*}
\llbracket \text{nat} \rrbracket & \overset{\text{def}}{=} \mathbb{N}_\bot \quad \text{(flat domain)} \\
\llbracket \text{bool} \rrbracket & \overset{\text{def}}{=} \mathbb{B}_\bot \quad \text{(flat domain)} \\
\llbracket \tau \rightarrow \tau' \rrbracket & \overset{\text{def}}{=} \llbracket \tau \rrbracket \rightarrow \llbracket \tau' \rrbracket \quad \text{(function domain)}.
\end{align*}
\]

where \(\mathbb{N} = \{0, 1, 2, \ldots\}\) and \(\mathbb{B} = \{\text{true}, \text{false}\}\).
Denotational semantics of PCF type environments

\[ [\Gamma] \overset{\text{def}}{=} \prod_{x \in \text{dom}(\Gamma)} [\Gamma(x)] \quad (\Gamma\text{-environments}) \]
Denotational semantics of PCF type environments

\[
[\Gamma] \overset{\text{def}}{=} \prod_{x \in \text{dom}(\Gamma)} [\Gamma(x)] \quad (\Gamma\text{-environments})
\]

\[
= \text{the domain of partial functions } \rho \text{ from variables to domains such that } \text{dom}(\rho) = \text{dom}(\Gamma) \text{ and } \rho(x) \in [\Gamma(x)] \text{ for all } x \in \text{dom}(\Gamma)
\]
Denotational semantics of PCF type environments

\[ [\Gamma] \overset{\text{def}}{=} \prod_{x \in \text{dom}(\Gamma)} [\Gamma(x)] \quad (\Gamma-\text{environments}) \]

\[ = \text{the domain of partial functions } \rho \text{ from variables to domains such that } \text{dom}(\rho) = \text{dom}(\Gamma) \text{ and } \rho(x) \in [\Gamma(x)] \text{ for all } x \in \text{dom}(\Gamma) \]

Example:

1. For the empty type environment \( \emptyset \),

\[ [\emptyset] = \{ \bot \} \]

where \( \bot \) denotes the unique partial function with \( \text{dom}(\bot) = \emptyset \).
2. $\llbracket \langle x \mapsto \tau \rangle \rrbracket = (\{ x \} \to \llbracket \tau \rrbracket)$
2. $\llbracket \langle x \mapsto \tau \rangle \rrbracket = (\{ x \} \rightarrow [\tau]) \simeq [\tau]$
2. $\llbracket \langle x \mapsto \tau \rangle \rrbracket = (\{ x \} \to \llbracket \tau \rrbracket) \cong \llbracket \tau \rrbracket$

3.

$\llbracket \langle x_1 \mapsto \tau_1, \ldots, x_n \mapsto \tau_n \rangle \rrbracket$

$\cong (\{ x_1 \} \to \llbracket \tau_1 \rrbracket) \times \ldots \times (\{ x_n \} \to \llbracket \tau_n \rrbracket)$

$\cong \llbracket \tau_1 \rrbracket \times \ldots \times \llbracket \tau_n \rrbracket$
Denotational semantics of PCF terms, I

\[
\text{⟦Γ ⊢ 0⟧}(\rho) \overset{\text{def}}{=} 0 \in [\text{nat}]
\]

\[
\text{⟦Γ ⊢ true⟧}(\rho) \overset{\text{def}}{=} \text{true} \in [\text{bool}]
\]

\[
\text{⟦Γ ⊢ false⟧}(\rho) \overset{\text{def}}{=} \text{false} \in [\text{bool}]
\]
Denotational semantics of PCF terms, I

\[ [\Gamma \vdash 0](\rho) \overset{\text{def}}{=} 0 \in [\text{nat}] \]

\[ [\Gamma \vdash \text{true}](\rho) \overset{\text{def}}{=} \text{true} \in [\text{bool}] \]

\[ [\Gamma \vdash \text{false}](\rho) \overset{\text{def}}{=} \text{false} \in [\text{bool}] \]

\[ [\Gamma \vdash x](\rho) \overset{\text{def}}{=} \rho(x) \in [\Gamma(x)] \quad (x \in \text{dom}(\Gamma)) \]
\[ [\Gamma \vdash \text{succ}(M)](\rho) \]

\[
= \begin{cases} 
[\Gamma \vdash M](\rho) + 1 & \text{if } [\Gamma \vdash M](\rho) \neq \bot \\
\bot & \text{if } [\Gamma \vdash M](\rho) = \bot
\end{cases}
\]
Denotational semantics of PCF terms, II

\[
\begin{align*}
\llbracket \Gamma \vdash \text{succ}(M) \rrbracket(\rho) & \overset{\text{def}}{=} \begin{cases} 
\llbracket \Gamma \vdash M \rrbracket(\rho) + 1 & \text{if } \llbracket \Gamma \vdash M \rrbracket(\rho) \neq \bot \\
\bot & \text{if } \llbracket \Gamma \vdash M \rrbracket(\rho) = \bot
\end{cases} \\
\llbracket \Gamma \vdash \text{pred}(M) \rrbracket(\rho) & \overset{\text{def}}{=} \begin{cases} 
\llbracket \Gamma \vdash M \rrbracket(\rho) - 1 & \text{if } \llbracket \Gamma \vdash M \rrbracket(\rho) > 0 \\
\bot & \text{if } \llbracket \Gamma \vdash M \rrbracket(\rho) = 0, \bot
\end{cases}
\end{align*}
\]
Denotational semantics of PCF terms, II

\[ \llbracket \Gamma \vdash \text{succ}(M) \rrbracket(\rho) \]
\[ \overset{\text{def}}{=} \begin{cases} \llbracket \Gamma \vdash M \rrbracket(\rho) + 1 & \text{if } \llbracket \Gamma \vdash M \rrbracket(\rho) \neq \bot \\ \bot & \text{if } \llbracket \Gamma \vdash M \rrbracket(\rho) = \bot \end{cases} \]

\[ \llbracket \Gamma \vdash \text{pred}(M) \rrbracket(\rho) \]
\[ \overset{\text{def}}{=} \begin{cases} \llbracket \Gamma \vdash M \rrbracket(\rho) - 1 & \text{if } \llbracket \Gamma \vdash M \rrbracket(\rho) > 0 \\ \bot & \text{if } \llbracket \Gamma \vdash M \rrbracket(\rho) = 0, \bot \end{cases} \]

\[ \llbracket \Gamma \vdash \text{zero}(M) \rrbracket(\rho) \overset{\text{def}}{=} \begin{cases} \text{true} & \text{if } \llbracket \Gamma \vdash M \rrbracket(\rho) = 0 \\ \text{false} & \text{if } \llbracket \Gamma \vdash M \rrbracket(\rho) > 0 \\ \bot & \text{if } \llbracket \Gamma \vdash M \rrbracket(\rho) = \bot \end{cases} \]
\[[\Gamma \vdash M_1 \text{ then } M_2 \text{ else } M_3](\rho)\]

\[\overset{\text{def}}{=} \begin{cases} 
\[\Gamma \vdash M_2\](\rho) & \text{if } \[\Gamma \vdash M_1\](\rho) = \text{true} \\
\[\Gamma \vdash M_3\](\rho) & \text{if } \[\Gamma \vdash M_1\](\rho) = \text{false} \\
\bot & \text{if } \[\Gamma \vdash M_1\](\rho) = \bot
\end{cases}\]
Denotational semantics of PCF terms, III

\[
\begin{align*}
\llbracket \Gamma \vdash \textbf{if } M_1 \textbf{ then } M_2 \textbf{ else } M_3 \rrbracket (\rho) & \overset{\text{def}}{=} \\
&= \begin{cases} 
\llbracket \Gamma \vdash M_2 \rrbracket (\rho) & \text{if } \llbracket \Gamma \vdash M_1 \rrbracket (\rho) = \text{true} \\
\llbracket \Gamma \vdash M_3 \rrbracket (\rho) & \text{if } \llbracket \Gamma \vdash M_1 \rrbracket (\rho) = \text{false} \\
\bot & \text{if } \llbracket \Gamma \vdash M_1 \rrbracket (\rho) = \bot
\end{cases}
\end{align*}
\]

\[
\llbracket \Gamma \vdash M_1 M_2 \rrbracket (\rho) \overset{\text{def}}{=} (\llbracket \Gamma \vdash M_1 \rrbracket (\rho)) (\llbracket \Gamma \vdash M_2 \rrbracket (\rho))
\]
Denotational semantics of PCF terms, IV

\[ [\Gamma \vdash \texttt{fn } x : \tau . M](\rho) \]
\[ \overset{\text{def}}{=} \lambda d \in [\tau] . [\Gamma[x \mapsto \tau] \vdash M](\rho[x \mapsto d]) \quad (x \notin \text{dom}(\Gamma)) \]

\textbf{NB:} \( \rho[x \mapsto d] \in [\Gamma[x \mapsto \tau]] \) is the function mapping \( x \) to \( d \in [\tau] \) and otherwise acting like \( \rho \).
Denotational semantics of PCF terms, V

\[
\left[\Gamma \vdash \text{fix}(M)\right](\rho) \overset{\text{def}}{=} \text{fix}\left(\left[\Gamma \vdash M\right](\rho)\right)
\]

Recall that \text{fix} is the function assigning least fixed points to continuous functions.
Denotational semantics of PCF

**Proposition.** For all typing judgements $\Gamma \vdash M : \tau$, the denotation

$$[\Gamma \vdash M] : [\Gamma] \rightarrow [\tau]$$

is a well-defined continuous function.
Denotations of closed terms

For a closed term $M \in \text{PCF}_\tau$, we get

$$[[\emptyset \vdash M] : [[\emptyset]] \rightarrow [[\tau]]]$$

and, since $[[\emptyset]] = \{ \bot \}$, we have

$$[[M]] \overset{\text{def}}{=} [[\emptyset \vdash M]](\bot) \in [[\tau]] \quad (M \in \text{PCF}_\tau)$$
Compositionality

**Proposition.** For all typing judgements $\Gamma \vdash M : \tau$ and $\Gamma \vdash M' : \tau$, and all contexts $C[-]$ such that $\Gamma' \vdash C[M] : \tau'$ and $\Gamma' \vdash C[M'] : \tau'$,

if $\llbracket\Gamma \vdash M\rrbracket = \llbracket\Gamma \vdash M'\rrbracket : \llbracket\Gamma\rrbracket \rightarrow \llbracket\tau\rrbracket$

then $\llbracket\Gamma' \vdash C[M]\rrbracket = \llbracket\Gamma' \vdash C[M]\rrbracket : \llbracket\Gamma'\rrbracket \rightarrow \llbracket\tau'\rrbracket$
Soundness

Proposition. For all closed terms $M, V \in \text{PCF}_\tau$, 

if $M \Downarrow_\tau V$ then $[M] = [V] \in [\tau]$.
Proposition. Suppose that $\Gamma \vdash M : \tau$ and that $\Gamma[x \mapsto \tau] \vdash M' : \tau'$, so that we also have $\Gamma \vdash M'[M/x] : \tau'$. Then,

$$[[\Gamma \vdash M'[M/x]]](\rho) = [[\Gamma[x \mapsto \tau] \vdash M']](\rho[x \mapsto [[\Gamma \vdash M]]])$$

for all $\rho \in [[\Gamma]]$. 
Substitution property

**Proposition.** Suppose that $\Gamma \vdash M : \tau$ and that $\Gamma [x \mapsto \tau] \vdash M' : \tau'$, so that we also have $\Gamma \vdash M'[M/x] : \tau'$. Then,

$$\left[ \left[ \Gamma \vdash M'[M/x] \right] \right] (\rho) = \left[ \Gamma [x \mapsto \tau] \vdash M' \right] (\rho [x \mapsto \left[ \Gamma \vdash M \right]])$$

for all $\rho \in \left[ \Gamma \right]$.

In particular when $\Gamma = \emptyset$, $\left[ \langle x \mapsto \tau \rangle \vdash M' \right] : \left[ \tau \right] \rightarrow \left[ \tau' \right]$ and

$$\left[ M'[M/x] \right] = \left[ \langle x \mapsto \tau \rangle \vdash M' \right] (\left[ M \right])$$
Topic 7

Relating Denotational and Operational Semantics
Adequacy

For any closed PCF terms $M$ and $V$ of ground type $\gamma \in \{\text{nat}, \text{bool}\}$ with $V$ a value

$$[M] = [V] \in [\gamma] \implies M \Downarrow_\gamma V.$$
For any closed PCF terms $M$ and $V$ of ground type $\gamma \in \{\text{nat}, \text{bool}\}$ with $V$ a value

$$[M] = [V] \in [\gamma] \implies M \Downarrow_\gamma V.$$
Adequacy

For any closed PCF terms $M$ and $V$ of ground type $\gamma \in \{\text{nat, bool}\}$ with $V$ a value

$[M] = [V] \in [\gamma] \Rightarrow M \downarrow_\gamma V$.

NB. Adequacy does not hold at function types:

$[\text{fn } x : \tau. (\text{fn } y : \tau. y) x] = [\text{fn } x : \tau. x] : [\tau] \rightarrow [\tau]$
Adequacy

For any closed PCF terms $M$ and $V$ of ground type $\gamma \in \{nat, bool\}$ with $V$ a value

$$[M] = [V] \in [\gamma] \implies M \Downarrow_\gamma V.$$ 

NB. Adequacy does not hold at function types:

$$[[fn\ x:\ \tau.\ (fn\ y:\ \tau.\ y)\ x]] = [[fn\ x:\ \tau.\ x]] : [[\tau]] \to [[\tau]]$$

but

$$fn\ x:\ \tau.\ (fn\ y:\ \tau.\ y)\ x \not\Downarrow_{\tau\to\tau} fn\ x:\ \tau.\ x$$
Adequacy proof idea
Adequacy proof idea

1. We cannot proceed to prove the adequacy statement by a straightforward induction on the structure of terms.

- Consider $M$ to be $M_1 M_2$, $\text{fix}(M')$. 
Adequacy proof idea

1. We cannot proceed to prove the adequacy statement by a straightforward induction on the structure of terms.
   
   ▶ Consider $M$ to be $M_1 M_2, \text{fix}(M')$.

2. So we proceed to prove a stronger statement that applies to terms of arbitrary types and implies adequacy.
Adequacy proof idea

1. We cannot proceed to prove the adequacy statement by a straightforward induction on the structure of terms.

   Consider $M$ to be $M_1 M_2, \text{fix}(M')$.

2. So we proceed to prove a stronger statement that applies to terms of arbitrary types and implies adequacy.

   This statement roughly takes the form:

   $$\llbracket M \rrbracket \triangleleft_\tau M$$ for all types $\tau$ and all $M \in \text{PCF}_\tau$

   where the formal approximation relations

   $$\triangleleft_\tau \subseteq \llbracket \tau \rrbracket \times \text{PCF}_\tau$$

   are logically chosen to allow a proof by induction.
Requirements on the formal approximation relations, I

We want that, for \( \gamma \in \{ \text{nat}, \text{bool} \} \),

\[
\llbracket M \rrbracket \mathrel{\triangleleft_\gamma} M \text{ implies } \forall V (\llbracket M \rrbracket = \llbracket V \rrbracket \implies M \Downarrow_\gamma V)
\]

\textbf{adequacy}
Definition of \( d \triangleleft_\gamma M \) (\( d \in \llbracket \gamma \rrbracket \), \( M \in \text{PCF}_\gamma \)) for \( \gamma \in \{\text{nat}, \text{bool}\} \)

\[
n \triangleleft_{\text{nat}} M \overset{\text{def}}{\iff} (n \in \mathbb{N} \Rightarrow M \downarrow_{\text{nat}} \text{succ}^n(0))
\]

\[
b \triangleleft_{\text{bool}} M \overset{\text{def}}{\iff} (b = \text{true} \Rightarrow M \downarrow_{\text{bool}} \text{true}) \\
\& (b = \text{false} \Rightarrow M \downarrow_{\text{bool}} \text{false})
\]
Proof of: $\left[ M \right] \triangleleft_\gamma M$ implies adequacy

Case $\gamma = \text{nat}$.

$$\left[ M \right] = \left[ V \right]$$

$$\Rightarrow \left[ M \right] = \left[ \text{succ}^n(0) \right]$$ for some $n \in \mathbb{N}$

$$\Rightarrow n = \left[ M \right] \triangleleft_\gamma M$$

$$\Rightarrow M \downarrow \text{succ}^n(0)$$ by definition of $\triangleleft_{\text{nat}}$

Case $\gamma = \text{bool}$ is similar.
Requirements on the formal approximation relations, II

We want to be able to proceed by induction.

Consider the case $M = M_1 M_2$.

\[ \sim logical \ definition \]
Definition of $f \triangleleft_{\tau \rightarrow \tau'} M$ ($f \in ([\tau] \rightarrow [\tau'])$, $M \in \text{PCF}_{\tau \rightarrow \tau'}$)
Definition of

\[ f \triangleleft_{\tau \to \tau'} M \quad (f \in ([\tau] \to [\tau']), M \in \text{PCF}_{\tau \to \tau'}) \]

\[ f \triangleleft_{\tau \to \tau'} M \]

\[ \overset{\text{def}}{\iff} \forall x \in [\tau], N \in \text{PCF}_\tau \]

\[ (x \triangleleft_\tau N \Rightarrow f(x) \triangleleft_{\tau'} M N) \]
Requirements on the formal approximation relations, III

We want to be able to proceed by induction.

Consider the case $M = \text{fix}(M')$.  

$\sim$ admissibility property
Admissibility property

**Lemma.** For all types $\tau$ and $M \in \text{PCF}_\tau$, the set

$$\{ d \in [\tau] \mid d \triangleleft_\tau M \}$$

is an admissible subset of $[\tau]$. 
Lemma. For all types $\tau$, elements $d, d' \in \llbracket \tau \rrbracket$, and terms $M, N, V \in \text{PCF}_\tau$,

1. If $d \sqsubseteq d'$ and $d' \prec_\tau M$ then $d \prec_\tau M$.

2. If $d \prec_\tau M$ and $\forall V \ (M \Downarrow_\tau V \implies N \Downarrow_\tau V)$ then $d \prec_\tau N$. 

We want to be able to proceed by induction.

Consider the case $M = \text{fn } x : \tau . M'$. 

$\sim \textit{substitutivity} \text{ property for open terms}$
**Fundamental property**

**Theorem.** For all \( \Gamma = \langle x_1 \mapsto \tau_1, \ldots, x_n \mapsto \tau_n \rangle \) and all \( \Gamma \vdash M : \tau \), if \( d_1 \triangleleft_{\tau_1} M_1, \ldots, d_n \triangleleft_{\tau_n} M_n \) then

\[
[\Gamma \vdash M][x_1 \mapsto d_1, \ldots, x_n \mapsto d_n] \triangleleft_{\tau} M[M_1/x_1, \ldots, M_n/x_n].
\]
Theorem. For all $\Gamma = \langle x_1 \mapsto \tau_1, \ldots, x_n \mapsto \tau_n \rangle$ and all $\Gamma \vdash M : \tau$, if $d_1 \triangleleft_{\tau_1} M_1, \ldots, d_n \triangleleft_{\tau_n} M_n$ then 
\[
\llbracket \Gamma \vdash M \rrbracket [x_1 \mapsto d_1, \ldots, x_n \mapsto d_n] \triangleleft_{\tau} M[M_1/x_1, \ldots, M_n/x_n].
\]

NB. The case $\Gamma = \emptyset$ reduces to 
\[
\llbracket M \rrbracket \triangleleft_{\tau} M
\]
for all $M \in \text{PCF}_{\tau}$. 
Fundamental property of the relations $\sqsubseteq_\tau$

**Proposition.** If $\Gamma \vdash M : \tau$ is a valid PCF typing, then for all $\Gamma$-environments $\rho$ and all $\Gamma$-substitutions $\sigma$

\[ \rho \sqsubseteq_\Gamma \sigma \Rightarrow \left[ \Gamma \vdash M \right](\rho) \sqsubseteq_\tau M[\sigma] \]

- $\rho \sqsubseteq_\Gamma \sigma$ means that $\rho(x) \sqsubseteq_{\Gamma(x)} \sigma(x)$ holds for each $x \in \text{dom}(\Gamma)$.
- $M[\sigma]$ is the PCF term resulting from the simultaneous substitution of $\sigma(x)$ for $x$ in $M$, each $x \in \text{dom}(\Gamma)$. 
Contextual preorder between PCF terms

Given PCF terms $M_1, M_2$, PCF type $\tau$, and a type environment $\Gamma$, the relation $\Gamma \vdash M_1 \leq_{ctx} M_2 : \tau$ 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 $V \in \text{PCF}_\gamma$,

$$C[M_1] \Downarrow_\gamma V \implies C[M_2] \Downarrow_\gamma V.$$
Extensionality properties of $\leq_{ctx}$

At a ground type $\gamma \in \{\text{bool, nat}\}$,

$M_1 \leq_{ctx} M_2 : \gamma$ holds if and only if

$$\forall V \in \text{PCF}_\gamma (M_1 \Downarrow_\gamma V \implies M_2 \Downarrow_\gamma V).$$

At a function type $\tau \rightarrow \tau'$,

$M_1 \leq_{ctx} M_2 : \tau \rightarrow \tau'$ holds if and only if

$$\forall M \in \text{PCF}_\tau (M_1 \ M \leq_{ctx} M_2 \ M : \tau').$$
Topic 8

Full Abstraction
Proof principle

For all types $\tau$ and closed terms $M_1, M_2 \in \text{PCF}_\tau$,

\[ \left[ [M_1] \right] = \left[ [M_2] \right] \text{ in } \left[ [\tau] \right] \implies M_1 \simeq_{\text{ctx}} M_2 : \tau . \]

Hence, to prove

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

it suffices to establish

\[ \left[ [M_1] \right] = \left[ [M_2] \right] \text{ in } \left[ [\tau] \right] . \]
A denotational model is said to be *fully abstract* whenever denotational equality characterises contextual equivalence.
Full abstraction

A denotational model is said to be *fully abstract* whenever denotational equality characterises contextual equivalence.

▶ The domain model of PCF is *not fully abstract*.

In other words, there are contextually equivalent PCF terms with different denotations.
Failure of full abstraction, idea

We will construct two closed terms

\[ T_1, T_2 \in \text{PCF}(\text{bool} \to (\text{bool} \to \text{bool})) \to \text{bool} \]

such that

\[ T_1 \cong_{\text{ctx}} T_2 \]

and

\[ [T_1] \neq [T_2] \]
We achieve $T_1 \simeq_{ctx} T_2$ by making sure that

$$\forall M \in \text{PCF}_{\text{bool} \rightarrow (\text{bool} \rightarrow \text{bool})} \left( T_1 M \nvdash \text{bool} \land T_2 M \nvdash \text{bool} \right)$$
We achieve $T_1 \simeq_{\text{ctx}} T_2$ by making sure that

$$\forall M \in \text{PCF}_{\text{bool} \rightarrow (\text{bool} \rightarrow \text{bool})} \ (T_1 \ M \not\Downarrow_{\text{bool}} \& T_2 \ M \not\Downarrow_{\text{bool}})$$

Hence,

$$[T_1]([M]) = \bot = [T_2]([M])$$

for all $M \in \text{PCF}_{\text{bool} \rightarrow (\text{bool} \rightarrow \text{bool})}$. 
We achieve $T_1 \simeq_{\text{ctx}} T_2$ by making sure that

$$\forall M \in \text{PCF}_{\text{bool} \to (\text{bool} \to \text{bool})} \left( T_1 M \not\Downarrow_{\text{bool}} \land T_2 M \not\Downarrow_{\text{bool}} \right)$$

Hence,

$$\llbracket T_1 \rrbracket (\llbracket M \rrbracket) = \bot = \llbracket T_2 \rrbracket (\llbracket M \rrbracket)$$

for all $M \in \text{PCF}_{\text{bool} \to (\text{bool} \to \text{bool})}$.

We achieve $\llbracket T_1 \rrbracket \neq \llbracket T_2 \rrbracket$ by making sure that

$$\llbracket T_1 \rrbracket (\text{por}) \neq \llbracket T_2 \rrbracket (\text{por})$$

for some non-definable continuous function

$$\text{por} \in (\mathbb{B}_\bot \to (\mathbb{B}_\bot \to \mathbb{B}_\bot)) .$$
Parallel-or function

is the unique continuous function \( \text{por} : \mathbb{B}_\bot \to (\mathbb{B}_\bot \to \mathbb{B}_\bot) \) such that

\[
\begin{align*}
\text{por} \ true \ \bot &= true \\
\text{por} \ \bot \ true &= true \\
\text{por} \ false \ false &= false
\end{align*}
\]
Parallel-or function

is the unique continuous function \( \text{por} : \mathbb{B}_\bot \rightarrow (\mathbb{B}_\bot \rightarrow \mathbb{B}_\bot) \) such that

\[
\begin{align*}
\text{por} \text{ true } \bot &= \text{ true } \\
\text{por} \bot \text{ true} &= \text{ true } \\
\text{por} \text{ false } \text{ false} &= \text{ false }
\end{align*}
\]

In which case, it necessarily follows by monotonicity that

\[
\begin{align*}
\text{por} \text{ true } \text{ true} &= \text{ true } \\
\text{por} \text{ false } \bot &= \bot \\
\text{por} \text{ true } \text{ false} &= \text{ true } \\
\text{por} \text{ false } \text{ true} &= \text{ true } \\
\text{por} \bot \bot &= \bot
\end{align*}
\]
Undefinability of parallel-or

**Proposition.** *There is no closed PCF term*

\[ P : \text{bool} \rightarrow (\text{bool} \rightarrow \text{bool}) \]

*satisfying*

\[ [P] = \text{por} : \text{B}_\perp \rightarrow (\text{B}_\perp \rightarrow \text{B}_\perp) \].
Parallel-or test functions
Parallel-or test functions

For \( i = 1, 2 \) define

\[
T_i \overset{\text{def}}{=} \text{fn } f : \text{bool } \to (\text{bool } \to \text{bool}).
\]

\[
\text{if } (f \text{ true } \Omega) \text{ then }
\]

\[
\text{if } (f \text{ false } \text{true }) \text{ then }
\]

\[
\text{if } (f \text{ false } \text{false }) \text{ then } \Omega \text{ else } B_i
\]

\[
\text{else } \Omega
\]

\[
\text{else } \Omega
\]

where \( B_1 \overset{\text{def}}{=} \text{true}, B_2 \overset{\text{def}}{=} \text{false}, \)

and \( \Omega \overset{\text{def}}{=} \text{fix(fnx : bool . x)}. \)
Failure of full abstraction

Proposition.

\[ T_1 \cong_{\text{ctx}} T_2 : (\text{bool} \rightarrow (\text{bool} \rightarrow \text{bool})) \rightarrow \text{bool} \]

\[ [T_1] \neq [T_2] \in (\mathbb{B}_\bot \rightarrow (\mathbb{B}_\bot \rightarrow \mathbb{B}_\bot)) \rightarrow \mathbb{B}_\bot \]
PCF+por

Expressions

\[ M ::= \cdots \mid \text{por}(M, M) \]

Typing

\[ \begin{align*}
\Gamma \vdash M_1 : \text{bool} & \quad \Gamma \vdash M_2 : \text{bool} \\
\Gamma \vdash \text{por}(M_1, M_2) : \text{bool}
\end{align*} \]

Evaluation

\[ \begin{align*}
M_1 \Downarrow_{\text{bool}} \text{true} & \quad \text{por}(M_1, M_2) \Downarrow_{\text{bool}} \text{true} \\
M_2 \Downarrow_{\text{bool}} \text{true} & \quad \text{por}(M_1, M_2) \Downarrow_{\text{bool}} \text{true} \\
M_1 \Downarrow_{\text{bool}} \text{false} & \quad M_2 \Downarrow_{\text{bool}} \text{false} \\
\text{por}(M_1, M_2) \Downarrow_{\text{bool}} \text{false}
\end{align*} \]
Plotkin’s full abstraction result

The denotational semantics of PCF+por is given by extending that of PCF with the clause

\[
[\Gamma \vdash \text{por}(M_1, M_2)](\rho) \overset{\text{def}}{=} \text{por}([\Gamma \vdash M_1](\rho))([\Gamma \vdash M_2](\rho))
\]

This denotational semantics is fully abstract for contextual equivalence of PCF+por terms:

\[
\Gamma \vdash M_1 \simeq_{\text{ctx}} M_2 : \tau \iff [\Gamma \vdash M_1] = [\Gamma \vdash M_2].
\]