Denotational Semantics

Lectures for Part II CST 2021/22

Prof Marcelo Fiore

Course web page:

http://www.cl.cam.ac.uk/teaching/2122/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

$$\text{Syntax} \quad \xrightarrow{[\![-]\!]} \quad \text{Semantics}$$

$$P \quad \mapsto \quad [\![P]\!]$$

Basic idea of denotational semantics

$$\text{Syntax} \xrightarrow{[\![-]\!]} \text{Semantics}$$

$$\text{Recursive program} \mapsto \text{Partial recursive function}$$

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

Basic idea of denotational semantics

Syntax	$\xrightarrow{[\![-]\!]}$	Semantics
Recursive program	$\mapsto$	Partial recursive function
Boolean circuit	$\mapsto$	Boolean function
P	$\mapsto$	$[\![P]\!]$

Basic idea of denotational semantics

$$
\begin{array}{rcl}
\text{Syntax} & \xrightarrow{[\![-]\!]} & \text{Semantics} \\
\text{Recursive program} & \mapsto & \text{Partial recursive function} \\
\text{Boolean circuit} & \mapsto & \text{Boolean function} \\
P & \mapsto & [\![P]\!]
\end{array}
$$

Concerns:

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

Basic idea of denotational semantics

$$
\begin{array}{rcl}
\text{Syntax} & \xrightarrow{[\![-]\!]} & \text{Semantics} \\
\text{Recursive program} & \mapsto & \text{Partial recursive function} \\
\text{Boolean circuit} & \mapsto & \text{Boolean function} \\
P & \mapsto & [\![P]\!]
\end{array}
$$

Concerns:

- Abstract models (*i.e.* implementation/machine independent).

  $\rightsquigarrow$ Lectures 2, 3 and 4.

- Compositionality.

  $\rightsquigarrow$ Lectures 5 and 6.

Basic idea of denotational semantics

$$\begin{array}{rcl}
\text{Syntax} & \xrightarrow{[\![-]\!]} & \text{Semantics} \\
\text{Recursive program} & \mapsto & \text{Partial recursive function} \\
\text{Boolean circuit} & \mapsto & \text{Boolean function} \\
P & \mapsto & [\![P]\!]
\end{array}$$

Concerns:

- Abstract models (*i.e.* implementation/machine independent).

  $\rightsquigarrow$ Lectures 2, 3 and 4.

- Compositionality.

  $\rightsquigarrow$ Lectures 5 and 6.

- Relationship to computation (*e.g.* operational semantics).

  $\rightsquigarrow$ 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

$$A \in \mathbf{Aexp} \quad ::= \quad \underline{n} \mid L \mid A + A \mid \ \dots$$

where n ranges over *integers* and L over a specified set of *locations* $\mathbb{L}$

Boolean expressions

$$\begin{array}{rcl} B \in \mathbf{Bexp} & ::= & \mathbf{true} \mid \mathbf{false} \mid A = A \mid \ \dots \\ & \mid & \neg B \mid \ \dots \end{array}$$

Commands

$$\begin{array}{rcl} C \in \mathbf{Comm} & ::= & \mathbf{skip} \mid L := A \mid C ; C \\ & \mid & \mathbf{if}\ B\ \mathbf{then}\ C\ \mathbf{else}\ C \end{array}$$

Basic example of denotational semantics (II)

Semantic functions

$$\mathcal{A} : \quad \mathbf{Aexp} \rightarrow (State \rightarrow \mathbb{Z})$$

where

$$\mathbb{Z} \quad = \quad \{\ldots, -1, 0, 1, \ldots\}$$

$$State \quad = \quad (\mathbb{L} \rightarrow \mathbb{Z})$$

Basic example of denotational semantics (II)

Semantic functions

$$\begin{aligned} \mathcal{A} : &\quad \mathbf{Aexp} \to (State \to \mathbb{Z}) \\ \mathcal{B} : &\quad \mathbf{Bexp} \to (State \to \mathbb{B}) \end{aligned}$$

where

$$\begin{aligned} \mathbb{Z} &= \{ \ldots, -1, 0, 1, \ldots \} \\ \mathbb{B} &= \{ true, false \} \\ State &= (\mathbb{L} \to \mathbb{Z}) \end{aligned}$$

Basic example of denotational semantics (II)

Semantic functions

$$\begin{aligned}
\mathcal{A}: &\quad \mathbf{Aexp} \to (\mathit{State} \to \mathbb{Z}) \\
\mathcal{B}: &\quad \mathbf{Bexp} \to (\mathit{State} \to \mathbb{B}) \\
\mathcal{C}: &\quad \mathbf{Comm} \to (\mathit{State} \rightharpoonup \mathit{State})
\end{aligned}$$

where

$$\begin{aligned}
\mathbb{Z} &= \{\ldots, -1, 0, 1, \ldots\} \\
\mathbb{B} &= \{\mathit{true}, \mathit{false}\} \\
\mathit{State} &= (\mathbb{L} \to \mathbb{Z})
\end{aligned}$$

Basic example of denotational semantics (III)

Semantic function $\mathcal{A}$

$$\mathcal{A}[\![\underline{n}]\!] = \lambda s \in State.\, n$$

$$\mathcal{A}[\![L]\!] = \lambda s \in State.\, s(L)$$

$$\mathcal{A}[\![A_1 + A_2]\!] = \lambda s \in State.\, \mathcal{A}[\![A_1]\!](s) + \mathcal{A}[\![A_2]\!](s)$$

Basic example of denotational semantics (IV)

Semantic function $\mathcal{B}$

$$
\begin{array}{rcl}
\mathcal{B}[\![\mathbf{true}]\!] & = & \lambda s \in \mathit{State}.\ \mathit{true} \\
\mathcal{B}[\![\mathbf{false}]\!] & = & \lambda s \in \mathit{State}.\ \mathit{false} \\
\mathcal{B}[\![A_1 = A_2]\!] & = & \lambda s \in \mathit{State}.\ \mathit{eq}\big(\mathcal{A}[\![A_1]\!](s), \mathcal{A}[\![A_2]\!](s)\big)
\end{array}
$$

where $eq(a, a') = \begin{cases} \mathit{true} & \text{if } a = a' \\ \mathit{false} & \text{if } a \neq a' \end{cases}$

Basic example of denotational semantics (V)

Semantic function $\mathcal{C}$

$$[\![\mathbf{skip}]\!] \quad = \quad \lambda s \in State.\, s$$

NB: From now on the names of semantic functions are omitted!

A simple example of compositionality

Given partial functions $[\![C]\!], [\![C']\!] : State \rightharpoonup State$ and a function $[\![B]\!] : State \to \{true, false\}$, we can define

$$[\![\mathbf{if}\ B\ \mathbf{then}\ C\ \mathbf{else}\ C']\!] = \\ \lambda s \in State.\ if\big([\![B]\!](s), [\![C]\!](s), [\![C']\!](s)\big)$$

where

$$if(b, x, x') = \begin{cases} x & \text{if } b = true \\ x' & \text{if } b = false \end{cases}$$

Basic example of denotational semantics (VI)

Semantic function $\mathcal{C}$

$$[\![L := A]\!] \quad = \quad \lambda s \in State.\, \lambda \ell \in \mathbb{L}.\, \mathit{if}\big(\ell = L, [\![A]\!](s), s(\ell)\big)$$

Denotational semantics of sequential composition

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

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

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

Denotational semantics of sequential composition

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

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

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

Cf. operational semantics of sequential composition:

$$\frac{C, s \Downarrow s' \quad C', s' \Downarrow s''}{C; C', s \Downarrow s''} \; .$$

$[\![\mathbf{while}\ B\ \mathbf{do}\ C]\!]$

Fixed point property of $[\![\mathbf{while}\ B\ \mathbf{do}\ C]\!]$

$$[\![\mathbf{while}\ B\ \mathbf{do}\ C]\!] = f_{[\![B]\!],[\![C]\!]}([\![\mathbf{while}\ B\ \mathbf{do}\ C]\!])$$

where, for each $b : State \to \{true, false\}$ and $c : State \rightharpoonup State$, we define

$$f_{b,c} : (State \rightharpoonup State) \to (State \rightharpoonup State)$$

as

$$f_{b,c} = \lambda w \in (State \rightharpoonup State).\, \lambda s \in State.\, if\,\big(b(s), w(c(s)), s\big).$$

Fixed point property of $[\![\textbf{while } B \textbf{ do } C]\!]$

$$[\![\textbf{while } B \textbf{ do } C]\!] = f_{[\![B]\!],[\![C]\!]}([\![\textbf{while } B \textbf{ do } C]\!])$$

where, for each $b : \mathit{State} \to \{\mathit{true}, \mathit{false}\}$ and $c : \mathit{State} \rightharpoonup \mathit{State}$, we define

$$f_{b,c} : (\mathit{State} \rightharpoonup \mathit{State}) \to (\mathit{State} \rightharpoonup \mathit{State})$$

as

$$f_{b,c} = \lambda w \in (\mathit{State} \rightharpoonup \mathit{State}).\, \lambda s \in \mathit{State}.\, \mathit{if}\,\big(b(s), w(c(s)), s\big).$$

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

Approximating $[\![\mathbf{while}\ B\ \mathbf{do}\ C]\!]$

Approximating $[\![\mathbf{while}\ B\ \mathbf{do}\ C]\!]$

$$
\begin{array}{l}
{f_{[\![B]\!],[\![C]\!]}}^n(\bot) \\
\quad = \quad \lambda s \in \mathit{State}. \\
\qquad\qquad \begin{cases} [\![C]\!]^k(s) & \text{if } \exists\, 0 \leq k < n.\, [\![B]\!]([\![C]\!]^k(s)) = \mathit{false} \\ & \text{and } \forall\, 0 \leq i < k.\, [\![B]\!]([\![C]\!]^i(s)) = \mathit{true} \\ \\ \uparrow & \text{if } \forall\, 0 \leq i < n.\, [\![B]\!]([\![C]\!]^i(s)) = \mathit{true} \end{cases}
\end{array}
$$

$$D \stackrel{\text{def}}{=} (State \rightharpoonup State)$$

- **Partial order $\sqsubseteq$ on D:**

$w \sqsubseteq w'$ iff for all $s \in State$, if w is defined at s then so is w' and moreover $w(s) = w'(s)$.

iff the graph of w is included in the graph of w'.

- **Least element $\perp \in D$ w.r.t. $\sqsubseteq$:**

$\perp$ = totally undefined partial function

= partial function with empty graph

(satisfies $\perp \sqsubseteq w$, for all $w \in D$).

Topic 2

Least Fixed Points

Thesis

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 monotonic.

Partially ordered sets

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.

$$\frac{}{x \sqsubseteq x}$$

$$\frac{x \sqsubseteq y \qquad y \sqsubseteq z}{x \sqsubseteq z}$$

$$\frac{x \sqsubseteq y \qquad y \sqsubseteq x}{x = y}$$

Domain of partial functions, $X \rightharpoonup Y$

Domain of partial functions, $X \rightharpoonup Y$

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

Domain of partial functions, $X \rightharpoonup Y$

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

Partial order:

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

Monotonicity

- A function $f : D \to E$ between posets is monotone iff

$$\forall d, d' \in D.\ d \sqsubseteq d' \Rightarrow f(d) \sqsubseteq f(d').$$

$$\frac{x \sqsubseteq y}{f(x) \sqsubseteq f(y)} \quad (f \text{ monotone})$$

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

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

- 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

$$\boxed{\mathit{fix}(f)}$$

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

$$f(\mathit{fix}(f)) \sqsubseteq \mathit{fix}(f) \tag{lfp1}$$

$$\forall d \in D.\ f(d) \sqsubseteq d \;\Rightarrow\; \mathit{fix}(f) \sqsubseteq d. \tag{lfp2}$$

Proof principle

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

   For all $x \in D$, to prove that $\mathit{fix}(f) \sqsubseteq x$ 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 $\mathit{fix}(f) \in D$.

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

$$\frac{f(x) \sqsubseteq x}{\mathit{fix}(f) \sqsubseteq x}$$

Proof principle

1.

$$\frac{}{f(\mathit{fix}(f)) \sqsubseteq \mathit{fix}(f)}$$

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

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

$$\frac{f(x) \sqsubseteq x}{\mathit{fix}(f) \sqsubseteq x}$$

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.

Thesis*

$$\boxed{\text{All domains of computation are complete partial orders with a least element.}}$$

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

Thesis*

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, $\bigsqcup_{n \geq 0} d_n$:

$$\forall m \geq 0 \,.\, d_m \sqsubseteq \bigsqcup_{n \geq 0} d_n \tag{lub1}$$

$$\forall d \in D \,.\, (\forall m \geq 0 \,.\, d_m \sqsubseteq d) \;\Rightarrow\; \bigsqcup_{n \geq 0} d_n \sqsubseteq d. \tag{lub2}$$

A domain is a cpo that possesses a least element, $\perp$:

$$\forall d \in D \,.\, \perp \sqsubseteq d.$$

$$\frac{}{\bot \sqsubseteq x}$$

$$\frac{}{x_i \sqsubseteq \bigsqcup_{n \geq 0} x_n} \quad (i \geq 0 \text{ and } \langle x_n \rangle \text{ a chain})$$

$$\frac{\forall n \geq 0 \,.\, x_n \sqsubseteq x}{\bigsqcup_{n \geq 0} x_n \sqsubseteq x} \quad (\langle x_i \rangle \text{ a chain})$$

Domain of partial functions, $X \rightharpoonup Y$

Domain of partial functions, $X \rightharpoonup Y$

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

Domain of partial functions, $X \rightharpoonup Y$

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

Partial order:

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

Domain of partial functions, $X \rightharpoonup Y$

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

Partial order:

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

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

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

Domain of partial functions, $X \rightharpoonup Y$

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

Partial order:

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

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

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

Least element $\perp$ 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 $\bigsqcup_n d_n \sqsubseteq \bigsqcup_n e_n$.

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 $\bigsqcup_n d_n \ \sqsubseteq \ \bigsqcup_n e_n$.

$$\frac{\forall n \geq 0 \, . \, x_n \sqsubseteq y_n}{\bigsqcup_n x_n \sqsubseteq \bigsqcup_n y_n} \quad (\langle x_n \rangle \text{ and } \langle y_n \rangle \text{ chains})$$

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*

$$m \leq m' \ \& \ n \leq n' \ \Rightarrow \ d_{m,n} \sqsubseteq d_{m',n'}. \qquad (\dagger)$$

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*

$$m \leq m' \ \&\ n \leq n' \ \Rightarrow \ d_{m,n} \sqsubseteq d_{m',n'}. \qquad (\dagger)$$

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

$$f(\bigsqcup_{n \geq 0} d_n) = \bigsqcup_{n \geq 0} f(d_n) \quad \text{in } E.$$

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 $f(\perp) = \perp$.

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

$$\mathit{fix}(f) = \bigsqcup_{n \geq 0} f^n(\perp).$$

- Moreover, $\mathit{fix}(f)$ is a fixed point of f, *i.e.* satisfies $f\big(\mathit{fix}(f)\big) = \mathit{fix}(f)$, and hence is the least fixed point of f.

$[\![\textbf{while } B \textbf{ do } C]\!]$

$$[\![\textbf{while } B \textbf{ do } C]\!]$$

$$= \mathit{fix}(f_{[\![B]\!],[\![C]\!]})$$

$$= \bigsqcup_{n \geq 0} {f_{[\![B]\!],[\![C]\!]}}^{n}(\perp)$$

$$= \quad \lambda s \in \mathit{State}. \left\{ \begin{array}{ll} [\![C]\!]^k(s) & \text{if } k \geq 0 \text{ is such that } [\![B]\!]([\![C]\!]^k(s)) = \mathit{false} \\ & \text{and } [\![B]\!]([\![C]\!]^i(s)) = \mathit{true} \text{ for all } 0 \leq i < k \\ & \\ \text{undefined} & \text{if } [\![B]\!]([\![C]\!]^i(s)) = \mathit{true} \text{ for all } i \geq 0 \end{array} \right.$$

Topic 3

Constructions on Domains

Discrete cpo's and flat domains

For any set X, the relation of equality

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

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

$$x \sqsubseteq x' \stackrel{\text{def}}{\Leftrightarrow} x = x' \qquad (x, x' \in X)$$

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

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

$$d \sqsubseteq d' \stackrel{\text{def}}{\Leftrightarrow} (d = d') \vee (d = \perp) \qquad (d, d' \in X_\perp)$$

makes $(X_\perp, \sqsubseteq)$ into a domain (with least element $\perp$), 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

$$(d_1, d_2) \sqsubseteq (d_1', d_2') \overset{\text{def}}{\Leftrightarrow} d_1 \sqsubseteq_1 d_1' \ \& \ d_2 \sqsubseteq_2 d_2' \ .$$

$$\frac{(x_1, x_2) \sqsubseteq (y_1, y_2)}{x_1 \sqsubseteq_1 y_1 \qquad x_2 \sqsubseteq_2 y_2}$$

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 $(D_1, \sqsubseteq_1)$ and $(D_2, \sqsubseteq_2)$ are domains so is $(D_1 \times D_2, \sqsubseteq)$ and $\perp_{D_1 \times D_2} = (\perp_{D_1}, \perp_{D_2})$.

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:*

$$\forall d, d' \in D, e \in E.\, d \sqsubseteq d' \;\Rightarrow\; f(d, e) \sqsubseteq f(d', e)$$

$$\forall d \in D, e, e' \in E.\, e \sqsubseteq e' \;\Rightarrow\; f(d, e) \sqsubseteq f(d, e').$$

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:

$$\frac{x \sqsubseteq x' \qquad y \sqsubseteq y'}{f(x,y) \sqsubseteq f(x',y')} \quad (f \text{ monotone})$$

$$\frac{}{f(\bigsqcup_m x_m, \bigsqcup_n y_n) = \bigsqcup_k f(x_k, y_k)}$$

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

$$(D \to E) \stackrel{\text{def}}{=} \{f \mid f : D \to E \text{ is a } \textit{continuous} \text{ function}\}$$

and partial order: $f \sqsubseteq f' \stackrel{\text{def}}{\Leftrightarrow} \forall d \in D \,.\, f(d) \sqsubseteq_E f'(d)$.

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

$$(D \to E) \stackrel{\text{def}}{=} \{f \mid f : D \to E \text{ is a } \textit{continuous} \text{ function}\}$$

and partial order: $f \sqsubseteq f' \stackrel{\text{def}}{\Leftrightarrow} \forall d \in D \,.\, f(d) \sqsubseteq_E f'(d)$.

- A derived rule:

$$\frac{f \sqsubseteq_{(D \to E)} g \qquad x \sqsubseteq_D y}{f(x) \sqsubseteq g(y)}$$

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 $D \to E$ and $\perp_{D\to E}(d) = \perp_E$, 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:

$$\frac{}{(\bigsqcup_n f_n)(\bigsqcup_m x_m) = \bigsqcup_k f_k(x_k)}$$

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

Continuity of composition

For cpo's D, E, F, the composition function

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

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

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

is continuous.

Continuity of the fixpoint operator

Let D be a domain.

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

Proposition. *The function*

$$\mathit{fix} : (D \to D) \to D$$

is continuous.

Topic 4

Scott Induction

Scott's Fixed Point Induction Principle

Let $f : D \to D$ 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

$$\mathit{fix}(f) \in S \ ,$$

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

$$(\forall n \geq 0 \,.\, d_n \in S) \;\Rightarrow\; \Big(\bigsqcup_{n \geq 0} d_n\Big) \in S$$

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

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

of D is chain-closed.

Building chain-closed subsets (I)

Let D, E be cpos.

Basic relations:

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

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

  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 $f : D \to D$ be a continuous function.

$$\forall d \in D.\, f(d) \sqsubseteq d \implies \mathit{fix}(f) \sqsubseteq d$$

Example (I): Least pre-fixed point property

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

$$\forall d \in D.\, f(d) \sqsubseteq d \implies \mathit{fix}(f) \sqsubseteq d$$

Proof by Scott induction.

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

$$\begin{aligned} x \in {\downarrow}(d) \quad &\implies \quad x \sqsubseteq d \\ &\implies \quad f(x) \sqsubseteq f(d) \\ &\implies \quad f(x) \sqsubseteq d \\ &\implies \quad f(x) \in {\downarrow}(d) \end{aligned}$$

Hence,

$$\mathit{fix}(f) \in {\downarrow}(d) \ .$$

Building chain-closed subsets (II)

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,

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

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,

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

Proof by Scott induction.

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

Since

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

we have that

$$f(\mathit{fix}(g)) \sqsubseteq g(\mathit{fix}(g)) \ .$$

Building chain-closed subsets (III)

Logical operations:

- If $S, T \subseteq D$ are chain-closed subsets of D then

$$S \cup T \qquad \text{and} \qquad S \cap T$$

  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.

Example (III): Partial correctness

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

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

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

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

where $f : (State \rightharpoonup State) \to (State \rightharpoonup State)$ is given by

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

Proof by Scott induction.

We consider the admissible subset of $(State \rightharpoonup State)$ given by

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

and show that

$$w \in S \implies f(w) \in S \ .$$

Topic 5

PCF

PCF syntax

Types

$$\tau ::= nat \mid bool \mid \tau \to \tau$$

PCF syntax

Types

$$\tau ::= nat \mid bool \mid \tau \to \tau$$

Expressions

$$M \quad ::= \quad \mathbf{0} \mid \mathbf{succ}(M) \mid \mathbf{pred}(M)$$

PCF syntax

Types

$$\tau ::= nat \mid bool \mid \tau \to \tau$$

Expressions

$$\begin{array}{rcl} M & ::= & \mathbf{0} \mid \mathbf{succ}(M) \mid \mathbf{pred}(M) \\ & \mid & \mathbf{true} \mid \mathbf{false} \mid \mathbf{zero}(M) \end{array}$$

PCF syntax

Types

$$\tau ::= nat \mid bool \mid \tau \to \tau$$

Expressions

$$\begin{array}{rcl} M & ::= & \mathbf{0} \mid \mathbf{succ}(M) \mid \mathbf{pred}(M) \\ & \mid & \mathbf{true} \mid \mathbf{false} \mid \mathbf{zero}(M) \\ & \mid & x \mid \mathbf{if}\ M\ \mathbf{then}\ M\ \mathbf{else}\ M \end{array}$$

PCF syntax

Types

$$\tau ::= nat \mid bool \mid \tau \to \tau$$

Expressions

$$\begin{array}{rcl} M & ::= & \mathbf{0} \mid \mathbf{succ}(M) \mid \mathbf{pred}(M) \\ & \mid & \mathbf{true} \mid \mathbf{false} \mid \mathbf{zero}(M) \\ & \mid & x \mid \mathbf{if}\ M\ \mathbf{then}\ M\ \mathbf{else}\ M \\ & \mid & \mathbf{fn}\, x : \tau \,.\, M \mid M\, M \mid \mathbf{fix}(M) \end{array}$$

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

PCF syntax

Types

$$\tau ::= nat \mid bool \mid \tau \to \tau$$

Expressions

$$\begin{array}{rcl} M & ::= & \mathbf{0} \mid \mathbf{succ}(M) \mid \mathbf{pred}(M) \\ & \mid & \mathbf{true} \mid \mathbf{false} \mid \mathbf{zero}(M) \\ & \mid & x \mid \mathbf{if}\ M\ \mathbf{then}\ M\ \mathbf{else}\ M \\ & \mid & \mathbf{fn}\, x : \tau \,.\, M \mid M\ M \mid \mathbf{fix}(M) \end{array}$$

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

Technicality: We identify expressions up to α-conversion of bound variables (created by the $\mathbf{fn}$ expression-former): by definition a PCF term is an α-equivalence class of expressions.

PCF typing relation, $\Gamma \vdash M : \tau$

- Γ is a type environment, *i.e.* a finite partial function mapping variables to types (whose domain of definition is denoted $dom(\Gamma)$)
- M is a term
- τ is a type.

PCF typing relation, $\Gamma \vdash M : \tau$

- Γ is a type environment, *i.e.* a finite partial function mapping variables to types (whose domain of definition is denoted $dom(\Gamma)$)
- M is a term
- τ is a type.

Notation:

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

$\mathrm{PCF}_\tau \stackrel{\text{def}}{=} \{M \mid M : \tau\}$.

PCF typing relation (sample rules)

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

PCF typing relation (sample rules)

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

$$(:_{\mathrm{app}}) \quad \frac{\Gamma \vdash M_1 : \tau \to \tau' \quad \Gamma \vdash M_2 : \tau}{\Gamma \vdash M_1\, M_2 : \tau'}$$

PCF typing relation (sample rules)

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

$$(:_{\mathrm{app}}) \quad \frac{\Gamma \vdash M_1 : \tau \to \tau' \quad \Gamma \vdash M_2 : \tau}{\Gamma \vdash M_1\, M_2 : \tau'}$$

$$(:_{\mathrm{fix}}) \quad \frac{\Gamma \vdash M : \tau \to \tau}{\Gamma \vdash \mathbf{fix}(M) : \tau}$$

Partial recursive functions in PCF

- Primitive recursion.

$$\begin{cases} h(x,0) = f(x) \\ h(x,y+1) = g(x,y,h(x,y)) \end{cases}$$

Partial recursive functions in PCF

- Primitive recursion.

$$\begin{cases} h(x,0) = f(x) \\ h(x,y+1) = g(x,y,h(x,y)) \end{cases}$$

- 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

- τ is a PCF type
- $M, V \in \mathrm{PCF}_\tau$ are closed PCF terms of type τ
- V is a value,

$$V ::= \mathbf{0} \mid \mathbf{succ}(V) \mid \mathbf{true} \mid \mathbf{false} \mid \mathbf{fn}\, x : \tau \,.\, M.$$

PCF evaluation (sample rules)

$$(\Downarrow_{\mathrm{val}}) \quad V \Downarrow_\tau V \qquad (V \text{ a value of type } \tau)$$

PCF evaluation (sample rules)

$$(\Downarrow_{\mathrm{val}}) \quad V \Downarrow_\tau V \qquad (V \text{ a value of type } \tau)$$

$$(\Downarrow_{\mathrm{cbn}}) \quad \frac{M_1 \Downarrow_{\tau \to \tau'} \mathbf{fn}\, x : \tau \,.\, M_1' \qquad M_1'[M_2/x] \Downarrow_{\tau'} V}{M_1\, M_2 \Downarrow_{\tau'} V}$$

PCF evaluation (sample rules)

$$(\Downarrow_{\mathrm{val}}) \quad V \Downarrow_\tau V \qquad (V \text{ a value of type } \tau)$$

$$(\Downarrow_{\mathrm{cbn}}) \quad \frac{M_1 \Downarrow_{\tau \to \tau'} \mathbf{fn}\, x : \tau \,.\, M_1' \qquad M_1'[M_2/x] \Downarrow_{\tau'} V}{M_1\, M_2 \Downarrow_{\tau'} V}$$

$$(\Downarrow_{\mathrm{fix}}) \quad \frac{M\, \mathbf{fix}(M) \Downarrow_\tau V}{\mathbf{fix}(M) \Downarrow_\tau V}$$

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, M_2, PCF type τ, and a type environment Γ, the relation $\boxed{\Gamma \vdash M_1 \cong_{\mathrm{ctx}} M_2 : \tau}$ is defined to hold iff

- Both the typings $\Gamma \vdash M_1 : \tau$ and $\Gamma \vdash M_2 : \tau$ hold.
- For all PCF contexts $\mathcal{C}$ for which $\mathcal{C}[M_1]$ and $\mathcal{C}[M_2]$ are closed terms of type γ, *where* $\gamma = nat$ *or* $\gamma = bool$, and for all values $V : \gamma$,

$$\mathcal{C}[M_1] \Downarrow_\gamma V \;\Leftrightarrow\; \mathcal{C}[M_2] \Downarrow_\gamma V.$$

PCF denotational semantics — aims

PCF denotational semantics — aims

- PCF types $\tau \quad \mapsto \quad$ domains $[\![\tau]\!]$.

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 $\tau \quad \mapsto \quad$ domains $[\![\tau]\!]$.

- Closed PCF terms $M : \tau \quad \mapsto \quad$ elements $[\![M]\!] \in [\![\tau]\!]$.

  Denotations of open terms will be continuous functions.

- Compositionality.

  In particular: $[\![M]\!] = [\![M']\!] \; \Rightarrow \; [\![\mathcal{C}[M]]\!] = [\![\mathcal{C}[M']]\!]$.

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: $[\![M]\!] = [\![M']\!] \;\Rightarrow\; [\![\mathcal{C}[M]]\!] = [\![\mathcal{C}[M']]\!]$.

- Soundness.

  For any type τ, $M \Downarrow_\tau V \;\Rightarrow\; [\![M]\!] = [\![V]\!]$.

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: $[\![M]\!] = [\![M']\!] \ \Rightarrow \ [\![\mathcal{C}[M]]\!] = [\![\mathcal{C}[M']]\!]$.
- Soundness.

  For any type τ, $M \Downarrow_\tau V \ \Rightarrow \ [\![M]\!] = [\![V]\!]$.
- Adequacy.

  For $\tau = bool$ or nat, $[\![M]\!] = [\![V]\!] \in [\![\tau]\!] \implies M \Downarrow_\tau V$.

Theorem. *For all types* τ *and closed terms* $M_1, M_2 \in \mathrm{PCF}_\tau$, *if* $[\![M_1]\!]$ *and* $[\![M_2]\!]$ *are equal elements of the domain* $[\![\tau]\!]$, *then* $M_1 \cong_{\mathrm{ctx}} M_2 : \tau$.

Theorem. *For all types* τ *and closed terms* $M_1, M_2 \in \mathrm{PCF}_\tau$, *if* $[\![M_1]\!]$ *and* $[\![M_2]\!]$ *are equal elements of the domain* $[\![\tau]\!]$, *then* $M_1 \cong_{\mathrm{ctx}} M_2 : \tau$.

Proof.

$$\mathcal{C}[M_1] \Downarrow_{nat} V \Rightarrow [\![\mathcal{C}[M_1]]\!] = [\![V]\!] \quad \text{(soundness)}$$

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

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

and symmetrically. □

Proof principle

To prove

$$M_1 \cong_{\mathrm{ctx}} M_2 : \tau$$

it suffices to establish

$$[\![M_1]\!] = [\![M_2]\!] \text{ in } [\![\tau]\!]$$

Proof principle

To prove

$$M_1 \cong_{\mathrm{ctx}} M_2 : \tau$$

it suffices to establish

$$[\![M_1]\!] = [\![M_2]\!] \text{ in } [\![\tau]\!]$$

$\boxed{?}$ 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

$$[\![\Gamma \vdash M]\!] : [\![\Gamma]\!] \to [\![\tau]\!]$$

between domains.

Denotational semantics of PCF types

$$[\![nat]\!] \stackrel{\text{def}}{=} \mathbb{N}_\perp \qquad \text{(flat domain)}$$

$$[\![bool]\!] \stackrel{\text{def}}{=} \mathbb{B}_\perp \qquad \text{(flat domain)}$$

where $\mathbb{N} = \{0, 1, 2, \ldots\}$ and $\mathbb{B} = \{true, false\}$.

Denotational semantics of PCF types

$$[\![nat]\!] \stackrel{\text{def}}{=} \mathbb{N}_\bot \qquad \text{(flat domain)}$$

$$[\![bool]\!] \stackrel{\text{def}}{=} \mathbb{B}_\bot \qquad \text{(flat domain)}$$

$$[\![\tau \to \tau']\!] \stackrel{\text{def}}{=} [\![\tau]\!] \to [\![\tau']\!] \qquad \text{(function domain).}$$

where $\mathbb{N} = \{0, 1, 2, \dots\}$ and $\mathbb{B} = \{true, false\}$.

Denotational semantics of PCF type environments

$$[\![\Gamma]\!] \stackrel{\text{def}}{=} \prod_{x \in dom(\Gamma)} [\![\Gamma(x)]\!]$$ (Γ-environments)

Denotational semantics of PCF type environments

$$[\![\Gamma]\!] \stackrel{\text{def}}{=} \prod_{x \in dom(\Gamma)} [\![\Gamma(x)]\!] \qquad (\Gamma\text{-environments})$$

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

Denotational semantics of PCF type environments

$$[\![\Gamma]\!] \stackrel{\text{def}}{=} \prod_{x \in dom(\Gamma)} [\![\Gamma(x)]\!] \qquad (\Gamma\text{-environments})$$

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

Example:

1. For the empty type environment $\emptyset$,

$$[\![\emptyset]\!] = \{ \perp \}$$

where $\perp$ denotes the unique partial function with $dom(\perp) = \emptyset$.

2. $[\![\langle x \mapsto \tau \rangle]\!] \;=\; \big(\{\, x \,\} \to [\![\tau]\!]\big)$

2. $[\![\langle x \mapsto \tau \rangle]\!] = \big(\{\, x \,\} \to [\![\tau]\!]\big) \cong [\![\tau]\!]$

2. $[\![\langle x \mapsto \tau \rangle]\!] = (\{\, x \,\} \to [\![\tau]\!]) \cong [\![\tau]\!]$

3.

$$
\begin{aligned}
& [\![\langle x_1 \mapsto \tau_1, \ldots, x_n \mapsto \tau_n \rangle]\!] \\
& \quad \cong (\{\, x_1 \,\} \to [\![\tau_1]\!]) \times \ldots \times (\{\, x_n \,\} \to [\![\tau_n]\!]) \\
& \quad \cong [\![\tau_1]\!] \times \ldots \times [\![\tau_n]\!]
\end{aligned}
$$

Denotational semantics of PCF terms, I

$$[\![\Gamma \vdash \mathbf{0}]\!](\rho) \stackrel{\text{def}}{=} 0 \in [\![nat]\!]$$

$$[\![\Gamma \vdash \mathbf{true}]\!](\rho) \stackrel{\text{def}}{=} true \in [\![bool]\!]$$

$$[\![\Gamma \vdash \mathbf{false}]\!](\rho) \stackrel{\text{def}}{=} false \in [\![bool]\!]$$

Denotational semantics of PCF terms, I

$$[\![\Gamma \vdash \mathbf{0}]\!](\rho) \stackrel{\text{def}}{=} 0 \in [\![nat]\!]$$

$$[\![\Gamma \vdash \mathbf{true}]\!](\rho) \stackrel{\text{def}}{=} true \in [\![bool]\!]$$

$$[\![\Gamma \vdash \mathbf{false}]\!](\rho) \stackrel{\text{def}}{=} false \in [\![bool]\!]$$

$$[\![\Gamma \vdash x]\!](\rho) \stackrel{\text{def}}{=} \rho(x) \in [\![\Gamma(x)]\!] \qquad \big(x \in dom(\Gamma)\big)$$

Denotational semantics of PCF terms, II

$$[\![\Gamma \vdash \mathbf{succ}(M)]\!](\rho)$$

$$\stackrel{\text{def}}{=} \begin{cases} [\![\Gamma \vdash M]\!](\rho) + 1 & \text{if } [\![\Gamma \vdash M]\!](\rho) \neq \bot \\ \bot & \text{if } [\![\Gamma \vdash M]\!](\rho) = \bot \end{cases}$$

Denotational semantics of PCF terms, II

$$[\![\Gamma \vdash \mathbf{succ}(M)]\!](\rho)$$

$$\stackrel{\text{def}}{=} \begin{cases} [\![\Gamma \vdash M]\!](\rho) + 1 & \text{if } [\![\Gamma \vdash M]\!](\rho) \neq \bot \\ \bot & \text{if } [\![\Gamma \vdash M]\!](\rho) = \bot \end{cases}$$

$$[\![\Gamma \vdash \mathbf{pred}(M)]\!](\rho)$$

$$\stackrel{\text{def}}{=} \begin{cases} [\![\Gamma \vdash M]\!](\rho) - 1 & \text{if } [\![\Gamma \vdash M]\!](\rho) > 0 \\ \bot & \text{if } [\![\Gamma \vdash M]\!](\rho) = 0, \bot \end{cases}$$

Denotational semantics of PCF terms, II

$$[\![\Gamma \vdash \mathbf{succ}(M)]\!](\rho)$$

$$\stackrel{\text{def}}{=} \begin{cases} [\![\Gamma \vdash M]\!](\rho) + 1 & \text{if } [\![\Gamma \vdash M]\!](\rho) \neq \bot \\ \bot & \text{if } [\![\Gamma \vdash M]\!](\rho) = \bot \end{cases}$$

$$[\![\Gamma \vdash \mathbf{pred}(M)]\!](\rho)$$

$$\stackrel{\text{def}}{=} \begin{cases} [\![\Gamma \vdash M]\!](\rho) - 1 & \text{if } [\![\Gamma \vdash M]\!](\rho) > 0 \\ \bot & \text{if } [\![\Gamma \vdash M]\!](\rho) = 0, \bot \end{cases}$$

$$[\![\Gamma \vdash \mathbf{zero}(M)]\!](\rho) \stackrel{\text{def}}{=} \begin{cases} \mathit{true} & \text{if } [\![\Gamma \vdash M]\!](\rho) = 0 \\ \mathit{false} & \text{if } [\![\Gamma \vdash M]\!](\rho) > 0 \\ \bot & \text{if } [\![\Gamma \vdash M]\!](\rho) = \bot \end{cases}$$

Denotational semantics of PCF terms, III

$$[\![\Gamma \vdash \mathbf{if}\ M_1\ \mathbf{then}\ M_2\ \mathbf{else}\ M_3]\!](\rho)$$

$$\stackrel{\text{def}}{=} \begin{cases} [\![\Gamma \vdash M_2]\!](\rho) & \text{if } [\![\Gamma \vdash M_1]\!](\rho) = \mathit{true} \\ [\![\Gamma \vdash M_3]\!](\rho) & \text{if } [\![\Gamma \vdash M_1]\!](\rho) = \mathit{false} \\ \bot & \text{if } [\![\Gamma \vdash M_1]\!](\rho) = \bot \end{cases}$$

Denotational semantics of PCF terms, III

$$[\![\Gamma \vdash \mathbf{if}\ M_1\ \mathbf{then}\ M_2\ \mathbf{else}\ M_3]\!](\rho)$$

$$\stackrel{\text{def}}{=} \begin{cases} [\![\Gamma \vdash M_2]\!](\rho) & \text{if } [\![\Gamma \vdash M_1]\!](\rho) = \mathit{true} \\ [\![\Gamma \vdash M_3]\!](\rho) & \text{if } [\![\Gamma \vdash M_1]\!](\rho) = \mathit{false} \\ \perp & \text{if } [\![\Gamma \vdash M_1]\!](\rho) = \perp \end{cases}$$

$$[\![\Gamma \vdash M_1\ M_2]\!](\rho) \stackrel{\text{def}}{=} \big([\![\Gamma \vdash M_1]\!](\rho)\big)\ ([\![\Gamma \vdash M_2]\!](\rho))$$

Denotational semantics of PCF terms, IV

$$\llbracket \Gamma \vdash \mathbf{fn}\, x : \tau \,.\, M \rrbracket(\rho)$$

$$\stackrel{\text{def}}{=} \lambda d \in \llbracket \tau \rrbracket \,.\, \llbracket \Gamma[x \mapsto \tau] \vdash M \rrbracket(\rho[x \mapsto d]) \qquad (x \notin dom(\Gamma))$$

NB: $\rho[x \mapsto d] \in \llbracket \Gamma[x \mapsto \tau] \rrbracket$ is the function mapping x to $d \in \llbracket \tau \rrbracket$ and otherwise acting like ρ.

Denotational semantics of PCF terms, V

$$[\![\Gamma \vdash \mathbf{fix}(M)]\!](\rho) \stackrel{\text{def}}{=} \mathit{fix}([\![\Gamma \vdash M]\!](\rho))$$

Recall that 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*

$$[\![\Gamma \vdash M]\!] : [\![\Gamma]\!] \to [\![\tau]\!]$$

is a well-defined continous function.

Denotations of closed terms

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

$$[\![\emptyset \vdash M]\!] : [\![\emptyset]\!] \to [\![\tau]\!]$$

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

$$\boxed{[\![M]\!] \stackrel{\text{def}}{=} [\![\emptyset \vdash M]\!](\perp) \in [\![\tau]\!] \qquad (M \in \mathrm{PCF}_\tau)}$$

Compositionality

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

$$\textit{if}\ \ [\![\Gamma \vdash M]\!] = [\![\Gamma \vdash M']\!] : [\![\Gamma]\!] \to [\![\tau]\!]$$

$$\textit{then}\ \ [\![\Gamma' \vdash \mathcal{C}[M]]\!] = [\![\Gamma' \vdash \mathcal{C}[M]]\!] : [\![\Gamma']\!] \to [\![\tau']\!]$$

Soundness

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

$$\textit{if } M \Downarrow_\tau V \textit{ then } [\![M]\!] = [\![V]\!] \in [\![\tau]\!] \ .$$

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,

$$
\begin{aligned}
&[\![\Gamma \vdash M'[M/x]]\!](\rho) \\
&\qquad = [\![\Gamma[x \mapsto \tau] \vdash M']\!]\big(\rho\big[x \mapsto [\![\Gamma \vdash M]\!](\rho)\big]\big)
\end{aligned}
$$

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,

$$
\begin{aligned}
&[\![\Gamma \vdash M'[M/x]]\!](\rho) \\
&\qquad = [\![\Gamma[x \mapsto \tau] \vdash M']\!]\big(\rho\big[x \mapsto [\![\Gamma \vdash M]\!](\rho)\big]\big)
\end{aligned}
$$

for all $\rho \in [\![\Gamma]\!]$*.*

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

$$
[\![M'[M/x]]\!] = [\![\langle x \mapsto \tau \rangle \vdash M']\!]([\![M]\!])
$$

Topic 7

Relating Denotational and Operational Semantics

Adequacy

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

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

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

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:

$$[\![\mathbf{fn}\, x : \tau.\, (\mathbf{fn}\, y : \tau.\, y)\, x]\!] \quad = \quad [\![\mathbf{fn}\, x : \tau.\, x]\!] \quad : [\![\tau]\!] \to [\![\tau]\!]$$

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:

$$[\![\mathbf{fn}\, x : \tau.\, (\mathbf{fn}\, y : \tau.\, y)\, x]\!] \quad = \quad [\![\mathbf{fn}\, x : \tau.\, x]\!] \quad : [\![\tau]\!] \to [\![\tau]\!]$$

but

$$\mathbf{fn}\, x : \tau.\, (\mathbf{fn}\, y : \tau.\, y)\, x \not\Downarrow_{\tau\to\tau} \mathbf{fn}\, x : \tau.\, x$$

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 $M_1\, M_2$, $\mathbf{fix}(M')$.

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 $M_1\,M_2$, $\mathbf{fix}(M')$.

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 $M_1\, M_2$, $\mathbf{fix}(M')$.

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:

   $$\boxed{[\![M]\!] \lhd_\tau M \text{ for all types } \tau \text{ and all } M \in \mathrm{PCF}_\tau}$$

   where the *formal approximation relations*

   $$\lhd_\tau \subseteq [\![\tau]\!] \times \mathrm{PCF}_\tau$$

   are *logically* chosen to allow a proof by induction.

Requirements on the formal approximation relations, I

We want that, for $\gamma \in \{nat, bool\}$,

$$[\![M]\!] \lhd_\gamma M \text{ implies } \underbrace{\forall V\,([\![M]\!] = [\![V]\!] \implies M \Downarrow_\gamma V)}_{\text{adequacy}}$$

Definition of $d \lhd_\gamma M \ (d \in [\![\gamma]\!], M \in \mathrm{PCF}_\gamma)$ for $\gamma \in \{nat, bool\}$

$$n \lhd_{nat} M \quad \overset{\text{def}}{\Leftrightarrow} \quad \big(n \in \mathbb{N} \ \Rightarrow \ M \Downarrow_{nat} \mathbf{succ}^n(\mathbf{0})\big)$$

$$\begin{aligned} b \lhd_{bool} M \quad \overset{\text{def}}{\Leftrightarrow} \quad & (b = true \ \Rightarrow \ M \Downarrow_{bool} \mathbf{true}) \\ & \&\ (b = false \ \Rightarrow \ M \Downarrow_{bool} \mathbf{false}) \end{aligned}$$

Proof of: $[\![M]\!] \lhd_\gamma M$ implies adequacy

Case $\gamma = nat$.

$$
\begin{aligned}
& [\![M]\!] = [\![V]\!] \\
& \quad \Longrightarrow \; [\![M]\!] = [\![\mathbf{succ}^n(\mathbf{0})]\!] && \text{for some } n \in \mathbb{N} \\
& \quad \Longrightarrow \; n = [\![M]\!] \lhd_\gamma M \\
& \quad \Longrightarrow \; M \Downarrow \mathbf{succ}^n(\mathbf{0}) && \text{by definition of } \lhd_{nat}
\end{aligned}
$$

Case $\gamma = bool$ is similar.

Requirements on the formal approximation relations, II

We want to be able to proceed by induction.

- Consider the case $M = M_1\, M_2$.

$\leadsto$ *logical* definition

Definition of
$f \lhd_{\tau \to \tau'} M \ \big(f \in (\llbracket \tau \rrbracket \to \llbracket \tau' \rrbracket), M \in \mathrm{PCF}_{\tau \to \tau'}\big)$

Definition of $f \lhd_{\tau\to\tau'} M \ \left(f \in (\llbracket\tau\rrbracket \to \llbracket\tau'\rrbracket), M \in \mathrm{PCF}_{\tau\to\tau'}\right)$

$$
\begin{aligned}
& f \lhd_{\tau\to\tau'} M \\
& \quad \overset{\text{def}}{\Leftrightarrow} \quad \forall\, x \in \llbracket\tau\rrbracket, N \in \mathrm{PCF}_\tau \\
& \qquad\qquad (x \lhd_\tau N \ \Rightarrow \ f(x) \lhd_{\tau'} M\, N)
\end{aligned}
$$

Requirements on the formal approximation relations, III

We want to be able to proceed by induction.

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

$\rightsquigarrow$ *admissibility* property

Admissibility property

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

$$\{\, d \in [\![\tau]\!] \mid d \lhd_\tau M \,\}$$

is an admissible subset of $[\![\tau]\!]$*.*

Further properties

Lemma. *For all types* τ, *elements* $d, d' \in [\![\tau]\!]$, *and terms* $M, N, V \in \mathrm{PCF}_\tau$,

1. *If* $d \sqsubseteq d'$ *and* $d' \lhd_\tau M$ *then* $d \lhd_\tau M$.
2. *If* $d \lhd_\tau M$ *and* $\forall V\,(M \Downarrow_\tau V \implies N \Downarrow_\tau V)$ *then* $d \lhd_\tau N$.

Requirements on the formal approximation relations, IV

We want to be able to proceed by induction.

- Consider the case $M = \mathbf{fn}\, x : \tau \,.\, M'$.

  $\leadsto$ *substitutivity* 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* $d_1 \lhd_{\tau_1} M_1, \ldots, d_n \lhd_{\tau_n} M_n$ *then* $[\![\Gamma \vdash M]\!][x_1 \mapsto d_1, \ldots, x_n \mapsto d_n] \lhd_\tau M[M_1/x_1, \ldots, M_n/x_n]$.

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* $d_1 \lhd_{\tau_1} M_1, \ldots, d_n \lhd_{\tau_n} M_n$ *then* $[\![\Gamma \vdash M]\!][x_1 \mapsto d_1, \ldots, x_n \mapsto d_n] \lhd_\tau M[M_1/x_1, \ldots, M_n/x_n]$.

NB. The case $\Gamma = \emptyset$ reduces to

$$[\![M]\!] \lhd_\tau M$$

for all $M \in \mathrm{PCF}_\tau$.

Fundamental property of the relations $\lhd_\tau$

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

$$\rho \lhd_\Gamma \sigma \;\Rightarrow\; [\![\Gamma \vdash M]\!](\rho) \lhd_\tau M[\sigma]$$

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

Contextual preorder between PCF terms

Given PCF terms M_1, M_2, PCF type τ, and a type environment Γ, the relation $\boxed{\Gamma \vdash M_1 \leq_{\mathrm{ctx}} M_2 : \tau}$ is defined to hold iff

- Both the typings $\Gamma \vdash M_1 : \tau$ and $\Gamma \vdash M_2 : \tau$ hold.
- For all PCF contexts $\mathcal{C}$ for which $\mathcal{C}[M_1]$ and $\mathcal{C}[M_2]$ are closed terms of type γ, *where* $\gamma = nat$ *or* $\gamma = bool$, and for all values $V \in \mathrm{PCF}_\gamma$,

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

Extensionality properties of $\leq_{\mathrm{ctx}}$

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

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

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

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

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

$$\forall\, M \in \mathrm{PCF}_\tau\ (M_1\, M \leq_{\mathrm{ctx}} M_2\, M : \tau')\ .$$

Topic 8

Full Abstraction

Proof principle

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

$$[\![M_1]\!] = [\![M_2]\!] \text{ in } [\![\tau]\!] \implies M_1 \cong_{\mathrm{ctx}} M_2 : \tau \ .$$

Hence, to prove

$$M_1 \cong_{\mathrm{ctx}} M_2 : \tau$$

it suffices to establish

$$[\![M_1]\!] = [\![M_2]\!] \text{ in } [\![\tau]\!] \ .$$

Full abstraction

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

$$T_1, T_2 \in \mathrm{PCF}_{(bool \to (bool \to bool)) \to bool}$$

such that

$$T_1 \cong_{\mathrm{ctx}} T_2$$

and

$$[\![T_1]\!] \neq [\![T_2]\!]$$

- We achieve $T_1 \cong_{\mathrm{ctx}} T_2$ by making sure that

$$\forall\, M \in \mathrm{PCF}_{bool\to(bool\to bool)}\,(\,T_1\, M \not\Downarrow_{bool}\ \&\ T_2\, M \not\Downarrow_{bool}\,)$$

- We achieve $T_1 \cong_{\mathrm{ctx}} T_2$ by making sure that

$$\forall\, M \in \mathrm{PCF}_{bool \to (bool \to bool)}\, (\, T_1\, M \not\Downarrow_{bool}\ \&\ T_2\, M \not\Downarrow_{bool}\,)$$

Hence,

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

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

- We achieve $T_1 \cong_{\mathrm{ctx}} T_2$ by making sure that

$$\forall\, M \in \mathrm{PCF}_{bool\to(bool\to bool)}\,(\,T_1\, M \not\Downarrow_{bool}\ \&\ T_2\, M \not\Downarrow_{bool}\,)$$

  Hence,

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

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

- We achieve $[\![T_1]\!] \neq [\![T_2]\!]$ by making sure that

$$[\![T_1]\!](por) \neq [\![T_2]\!](por)$$

  for some *non-definable* continuous function

$$por \in (\mathbb{B}_\bot \to (\mathbb{B}_\bot \to \mathbb{B}_\bot))\ .$$

Parallel-or function

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

$$\begin{array}{lcl} por\ \ true\ \ \perp & = & true \\ por\ \ \perp\ \ true & = & true \\ por\ \ false\ \ false & = & false \end{array}$$

Parallel-or function

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

$$\begin{aligned} por\ true\ \perp &= true \\ por\ \perp\ true &= true \\ por\ false\ false &= false \end{aligned}$$

In which case, it necessarily follows by monotonicity that

$$\begin{aligned} por\ true\ true &= true & \qquad por\ false\ \perp &= \perp \\ por\ true\ false &= true & \qquad por\ \perp\ false &= \perp \\ por\ false\ true &= true & \qquad por\ \perp\ \perp &= \perp \end{aligned}$$

Undefinability of parallel-or

Proposition. *There is no closed PCF term*

$$P : bool \to (bool \to bool)$$

satisfying

$$[\![P]\!] = por : \mathbb{B}_\perp \to (\mathbb{B}_\perp \to \mathbb{B}_\perp) \ .$$

Parallel-or test functions

Parallel-or test functions

For $i = 1, 2$ define

$$
\begin{aligned}
T_i \stackrel{\text{def}}{=} \quad & \mathbf{fn}\, f : bool \to (bool \to bool)\,.\\
& \quad \mathbf{if}\ (f\ \mathbf{true}\ \Omega)\ \mathbf{then}\\
& \quad\quad \mathbf{if}\ (f\ \Omega\ \mathbf{true})\ \mathbf{then}\\
& \quad\quad\quad \mathbf{if}\ (f\ \mathbf{false}\ \mathbf{false})\ \mathbf{then}\ \Omega\ \mathbf{else}\ B_i\\
& \quad\quad \mathbf{else}\ \Omega\\
& \quad \mathbf{else}\ \Omega
\end{aligned}
$$

where $B_1 \stackrel{\text{def}}{=} \mathbf{true}$, $B_2 \stackrel{\text{def}}{=} \mathbf{false}$,
and $\Omega \stackrel{\text{def}}{=} \mathbf{fix}(\mathbf{fn}\, x : bool\,.\, x)$.

Failure of full abstraction

Proposition.

$$T_1 \cong_{\mathrm{ctx}} T_2 : (bool \to (bool \to bool)) \to bool$$

$$[\![T_1]\!] \neq [\![T_2]\!] \in (\mathbb{B}_\perp \to (\mathbb{B}_\perp \to \mathbb{B}_\perp)) \to \mathbb{B}_\perp$$

PCF+por

Expressions $\qquad M ::= \cdots \mid \mathbf{por}(M, M)$

Typing
$$\frac{\Gamma \vdash M_1 : bool \quad \Gamma \vdash M_2 : bool}{\Gamma \vdash \mathbf{por}(M_1, M_2) : bool}$$

Evaluation

$$\frac{M_1 \Downarrow_{bool} \mathbf{true}}{\mathbf{por}(M_1, M_2) \Downarrow_{bool} \mathbf{true}} \qquad \frac{M_2 \Downarrow_{bool} \mathbf{true}}{\mathbf{por}(M_1, M_2) \Downarrow_{bool} \mathbf{true}}$$

$$\frac{M_1 \Downarrow_{bool} \mathbf{false} \quad M_2 \Downarrow_{bool} \mathbf{false}}{\mathbf{por}(M_1, M_2) \Downarrow_{bool} \mathbf{false}}$$

Plotkin's full abstraction result

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

$$[\![\Gamma \vdash \mathbf{por}(M_1, M_2)]\!](\rho) \stackrel{\text{def}}{=} por\big([\![\Gamma \vdash M_1]\!](\rho)\big)\big([\![\Gamma \vdash M_2]\!](\rho)\big)$$

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

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