

Denotational Semantics

10 lectures for Part II CST 2014/15

Marcelo Fiore

Course web page:

<http://www.cl.cam.ac.uk/teaching/1415/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 $\xrightarrow{[[-]]}$ Semantics

$P \mapsto [[P]]$

Basic idea of denotational semantics

Syntax $\xrightarrow{\llbracket - \rrbracket}$ Semantics

Recursive program \mapsto Partial recursive function

$P \mapsto \llbracket P \rrbracket$

Basic idea of denotational semantics

Syntax $\xrightarrow{\llbracket - \rrbracket}$ Semantics

Recursive program \mapsto Partial recursive function

Boolean circuit \mapsto Boolean function

P \mapsto $\llbracket P \rrbracket$

Basic idea of denotational semantics

Syntax $\xrightarrow{\llbracket - \rrbracket}$ 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).
 \rightsquigarrow Lectures 2, 3 and 4.

Basic idea of denotational semantics

Syntax	$\xrightarrow{\llbracket - \rrbracket}$	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).
 \rightsquigarrow Lectures 2, 3 and 4.
- Compositionality.
 \rightsquigarrow Lectures 5 and 6.

Basic idea of denotational semantics

Syntax	$\xrightarrow{\llbracket - \rrbracket}$	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).
 \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**, $\llbracket P \rrbracket$ — 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

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

where n ranges over *integers* and

L over a specified set of *locations* \mathbb{L}

Boolean expressions

$B \in \mathbf{Bexp} ::= \mathbf{true} \mid \mathbf{false} \mid A = A \mid \dots$
 $\mid \neg B \mid \dots$

Commands

$C \in \mathbf{Comm} ::= \mathbf{skip} \mid L := A \mid C; C$
 $\mid \mathbf{if } B \mathbf{ then } C \mathbf{ else } C$

Basic example of denotational semantics (II)

Semantic functions

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

where

$$\mathbb{Z} = \{ \dots, -1, 0, 1, \dots \}$$

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

Basic example of denotational semantics (II)

Semantic functions

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

$$\mathcal{B} : \mathbf{Bexp} \rightarrow (State \rightarrow \mathbb{B})$$

where

$$\mathbb{Z} = \{ \dots, -1, 0, 1, \dots \}$$

$$\mathbb{B} = \{ true, false \}$$

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

Basic example of denotational semantics (II)

Semantic functions

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

$$\mathcal{B} : \mathbf{Bexp} \rightarrow (State \rightarrow \mathbb{B})$$

$$\mathcal{C} : \mathbf{Comm} \rightarrow (State \rightarrow State)$$

where

$$\mathbb{Z} = \{ \dots, -1, 0, 1, \dots \}$$

$$\mathbb{B} = \{ true, false \}$$

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

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}

$$\mathcal{B}[\mathbf{true}] = \lambda s \in State. true$$

$$\mathcal{B}[\mathbf{false}] = \lambda s \in State. false$$

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

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

Basic example of denotational semantics (V)

Semantic function \mathcal{C}

$$\llbracket \text{skip} \rrbracket = \lambda s \in \text{State}. s$$

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

A simple example of compositionality

Given partial functions $\llbracket C \rrbracket, \llbracket C' \rrbracket : State \rightarrow State$ and a function $\llbracket B \rrbracket : State \rightarrow \{true, false\}$, we can define

$\llbracket \text{if } B \text{ then } C \text{ else } C' \rrbracket =$

$$\lambda s \in State. \text{if} (\llbracket B \rrbracket(s), \llbracket C \rrbracket(s), \llbracket C' \rrbracket(s))$$

where

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

Basic example of denotational semantics (VI)

Semantic function \mathcal{C}

$$\llbracket L := A \rrbracket = \lambda s \in \text{State}. \lambda \ell \in \mathbb{L}. \text{if } (\ell = L, \llbracket A \rrbracket (s), s(\ell))$$

Denotational semantics of sequential composition

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

$$\llbracket C; C' \rrbracket = \llbracket C' \rrbracket \circ \llbracket C \rrbracket = \lambda s \in \text{State}. \llbracket C' \rrbracket (\llbracket C \rrbracket (s))$$

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

Denotational semantics of sequential composition

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

$$\llbracket C; C' \rrbracket = \llbracket C' \rrbracket \circ \llbracket C \rrbracket = \lambda s \in State. \llbracket C' \rrbracket (\llbracket C \rrbracket (s))$$

given by composition of the partial functions from states to states $\llbracket C \rrbracket, \llbracket C' \rrbracket : State \rightarrow State$ which are the denotations of the commands.

Cf. operational semantics of sequential composition:

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

[[while *B* do *C*]]

Fixed point property of [[while B do C]]

$$\llbracket \text{while } B \text{ do } C \rrbracket = f_{\llbracket B \rrbracket, \llbracket C \rrbracket}(\llbracket \text{while } B \text{ do } C \rrbracket)$$

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

as $f_{b,c} : (State \rightarrow State) \rightarrow (State \rightarrow State)$

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

Fixed point property of [[while B do C]]

$$\llbracket \text{while } B \text{ do } C \rrbracket = f_{\llbracket B \rrbracket, \llbracket C \rrbracket}(\llbracket \text{while } B \text{ do } C \rrbracket)$$

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

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

as

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

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

Approximating `[[while B do C]]`

Approximating $\llbracket \text{while } B \text{ do } C \rrbracket$

$$f_{\llbracket B \rrbracket, \llbracket C \rrbracket}^n(\perp)$$

$$= \lambda s \in \text{State}.$$

$$\left\{ \begin{array}{l} \llbracket C \rrbracket^k(s) \quad \text{if } \exists 0 \leq k < n. \llbracket B \rrbracket(\llbracket C \rrbracket^k(s)) = \text{false} \\ \quad \text{and } \forall 0 \leq i < k. \llbracket B \rrbracket(\llbracket C \rrbracket^i(s)) = \text{true} \\ \uparrow \quad \text{if } \forall 0 \leq i < n. \llbracket B \rrbracket(\llbracket C \rrbracket^i(s)) = \text{true} \end{array} \right.$$

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

- **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 mononotic.

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**.

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

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

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

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, $X \rightarrow Y$

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

Partial order:

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

Monotonicity

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

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

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

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

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

- 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 $f : D \rightarrow D$ 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

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

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

$$f(\text{fix}(f)) \sqsubseteq \text{fix}(f) \quad (\text{lfp1})$$

$$\forall d \in D. f(d) \sqsubseteq d \Rightarrow \text{fix}(f) \sqsubseteq d. \quad (\text{lfp2})$$

Proof principle

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

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

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

Proof principle

1.

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

2. Let D be a poset and let $f : D \rightarrow D$ 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 $f(x) \sqsubseteq x$.

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

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.

Thesis^{*}

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 $d_0 \sqsubseteq d_1 \sqsubseteq d_2 \sqsubseteq \dots$ have least upper bounds, $\bigsqcup_{n \geq 0} d_n$:

$$\forall m \geq 0 . d_m \sqsubseteq \bigsqcup_{n \geq 0} d_n \quad (\text{lub1})$$

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

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

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

$$\overline{\perp \sqsubseteq x}$$

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

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

Domain of partial functions, $X \rightharpoonup Y$

Domain of partial functions, $X \rightharpoonup Y$

Underlying set: all partial functions, f , with domain of definition $\text{dom}(f) \subseteq X$ 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:

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

Domain of partial functions, $X \rightarrow Y$

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

Partial order:

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

Lub of chain $f_0 \sqsubseteq f_1 \sqsubseteq f_2 \sqsubseteq \dots$ 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, $X \rightarrow Y$

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

Partial order:

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

Lub of chain $f_0 \sqsubseteq f_1 \sqsubseteq f_2 \sqsubseteq \dots$ 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 $d_0 \sqsubseteq d_1 \sqsubseteq \dots \sqsubseteq d_n \sqsubseteq \dots$ in D ,

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

for all $N \in \mathbb{N}$.

3. For every pair of chains $d_0 \sqsubseteq d_1 \sqsubseteq \dots \sqsubseteq d_n \sqsubseteq \dots$ and $e_0 \sqsubseteq e_1 \sqsubseteq \dots \sqsubseteq e_n \sqsubseteq \dots$ 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 $d_0 \sqsubseteq d_1 \sqsubseteq \dots \sqsubseteq d_n \sqsubseteq \dots$ and $e_0 \sqsubseteq e_1 \sqsubseteq \dots \sqsubseteq e_n \sqsubseteq \dots$ 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

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

Then

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

and

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

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

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

Then

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

and

$$\bigsqcup_{m \geq 0} d_{m,0} \sqsubseteq \bigsqcup_{m \geq 0} d_{m,1} \sqsubseteq \bigsqcup_{m \geq 0} d_{m,2} \sqsubseteq \dots$$

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 $d_0 \sqsubseteq d_1 \sqsubseteq \dots$ 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 $d_0 \sqsubseteq d_1 \sqsubseteq \dots$ in D , it is the case that

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

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

Tarski's Fixed Point Theorem

Let $f : D \rightarrow D$ be a continuous function on a domain D . Then

- f possesses a least pre-fixed point, given by

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

- 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 .

[[while B do C]]

[[while B do C]]

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

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

$$= \lambda s \in \text{State}.$$

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

Topic 3

Constructions on Domains

Discrete cpo's and flat domains

For any set X , the relation of equality

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

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

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

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

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

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

$$(d_1, d_2) \sqsubseteq (d'_1, d'_2) \stackrel{\text{def}}{\iff} d_1 \sqsubseteq_1 d'_1 \ \& \ d_2 \sqsubseteq_2 d'_2 .$$

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

Lubs of chains are calculated componentwise:

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

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 $f : (D \times E) \rightarrow F$ 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:

$$f\left(\bigsqcup_{m \geq 0} d_m, e\right) = \bigsqcup_{m \geq 0} f(d_m, e)$$

$$f\left(d, \bigsqcup_{n \geq 0} e_n\right) = \bigsqcup_{n \geq 0} f(d, e_n).$$

- A couple of derived rules:

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

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

Function cpo's and domains

Given cpo's (D, \sqsubseteq_D) and (E, \sqsubseteq_E) , the **function cpo** $(D \rightarrow E, \sqsubseteq)$ has underlying set

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

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

Function cpo's and domains

Given cpo's (D, \sqsubseteq_D) and (E, \sqsubseteq_E) , the **function cpo** $(D \rightarrow E, \sqsubseteq)$ has underlying set

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

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

- A derived rule:

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

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

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

If E is a domain, then so is $D \rightarrow E$ and $\perp_{D \rightarrow E}(d) = \perp_E$, all $d \in D$.

Continuity of composition

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

$$\circ : ((E \rightarrow F) \times (D \rightarrow E)) \longrightarrow (D \rightarrow F)$$

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

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

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, $fix(f) \in D$.

Proposition. *The function*

$$fix : (D \rightarrow D) \rightarrow D$$

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 $d_0 \sqsubseteq d_1 \sqsubseteq d_2 \sqsubseteq \dots$ 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 $d_0 \sqsubseteq d_1 \sqsubseteq d_2 \sqsubseteq \dots$ 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

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

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 $f : D \rightarrow D$ 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 $f : D \rightarrow D$ 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,

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

Hence,

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

Building chain-closed subsets (II)

Inverse image:

Let $f : D \rightarrow E$ 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 $f, g : D \rightarrow D$ be continuous functions such that $f \circ g \sqsubseteq g \circ f$. Then,

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

Example (II)

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

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

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

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

Building chain-closed subsets (III)

Logical operations:

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

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

are chain-closed subsets of D .

- If $\{S_i\}_{i \in I}$ is a family of chain-closed subsets of D indexed by a set I , then $\bigcap_{i \in I} S_i$ is a chain-closed subset of D .
- If a property $P(x, y)$ determines a chain-closed subset of $D \times E$, then the property $\forall x \in D. P(x, y)$ determines a chain-closed subset of E .

Example (III): Partial correctness

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

while $X > 0$ **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

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

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

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

Proof by Scott induction.

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

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

and show that

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

Topic 5

PCF

PCF syntax

Types

$$\tau ::= \text{nat} \mid \text{bool} \mid \tau \rightarrow \tau$$

PCF syntax

Types

$$\tau ::= \text{nat} \mid \text{bool} \mid \tau \rightarrow \tau$$

Expressions

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

PCF syntax

Types

$$\tau ::= \text{nat} \mid \text{bool} \mid \tau \rightarrow \tau$$

Expressions

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

PCF syntax

Types

$$\tau ::= \text{nat} \mid \text{bool} \mid \tau \rightarrow \tau$$

Expressions

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

PCF syntax

Types

$$\tau ::= \text{nat} \mid \text{bool} \mid \tau \rightarrow \tau$$

Expressions

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

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

PCF syntax

Types

$$\tau ::= \mathit{nat} \mid \mathit{bool} \mid \tau \rightarrow \tau$$

Expressions

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

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

Technicality: We identify expressions up to α -conversion of bound variables (created by the **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.

$PCF_{\tau} \stackrel{\text{def}}{=} \{M \mid M : \tau\}$.

PCF typing relation (sample rules)

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

PCF typing relation (sample rules)

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

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

PCF typing relation (sample rules)

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

$$(\cdot\text{app}) \quad \frac{\Gamma \vdash M_1 : \tau \rightarrow \tau' \quad \Gamma \vdash M_2 : \tau}{\Gamma \vdash M_1 M_2 : \tau'}$$

$$(\cdot\text{fix}) \quad \frac{\Gamma \vdash M : \tau \rightarrow \tau}{\Gamma \vdash \mathbf{fix}(M) : \tau}$$

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
- $M, V \in \text{PCF}_{\tau}$ 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)

$(\Downarrow_{\text{val}})$ $V \Downarrow_{\tau} V$ (V a value of type τ)

PCF evaluation (sample rules)

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

$$(\Downarrow_{\text{cbn}}) \quad \frac{M_1 \Downarrow_{\tau \rightarrow \tau'} \mathbf{fn} \ x : \tau . M'_1 \quad M'_1[M_2/x] \Downarrow_{\tau'} V}{M_1 M_2 \Downarrow_{\tau'} V}$$

PCF evaluation (sample rules)

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

$$(\Downarrow_{\text{cbn}}) \quad \frac{M_1 \Downarrow_{\tau \rightarrow \tau'} \mathbf{fn} \ x : \tau . M'_1 \quad M'_1[M_2/x] \Downarrow_{\tau'} V}{M_1 M_2 \Downarrow_{\tau'} V}$$

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

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 $\Gamma \vdash M_1 \cong_{\text{ctx}} M_2 : \tau$ is defined to hold iff

- Both the typings $\Gamma \vdash M_1 : \tau$ and $\Gamma \vdash M_2 : \tau$ hold.
- For all PCF contexts \mathcal{C} for which $\mathcal{C}[M_1]$ and $\mathcal{C}[M_2]$ are closed terms of type γ , where $\gamma = \text{nat}$ or $\gamma = \text{bool}$, 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 τ \mapsto 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 $\tau \mapsto$ domains $\llbracket \tau \rrbracket$.
- Closed PCF terms $M : \tau \mapsto$ elements $\llbracket M \rrbracket \in \llbracket \tau \rrbracket$.
Denotations of open terms will be continuous functions.
- **Compositionality**.
In particular: $\llbracket M \rrbracket = \llbracket M' \rrbracket \Rightarrow \llbracket \mathcal{C}[M] \rrbracket = \llbracket \mathcal{C}[M'] \rrbracket$.

PCF denotational semantics — aims

- PCF types $\tau \mapsto$ domains $\llbracket \tau \rrbracket$.
- Closed PCF terms $M : \tau \mapsto$ elements $\llbracket M \rrbracket \in \llbracket \tau \rrbracket$.
Denotations of open terms will be continuous functions.
- **Compositionality**.
In particular: $\llbracket M \rrbracket = \llbracket M' \rrbracket \Rightarrow \llbracket \mathcal{C}[M] \rrbracket = \llbracket \mathcal{C}[M'] \rrbracket$.
- **Soundness**.
For any type τ , $M \Downarrow_{\tau} V \Rightarrow \llbracket M \rrbracket = \llbracket V \rrbracket$.

PCF denotational semantics — aims

- PCF types $\tau \mapsto$ domains $\llbracket \tau \rrbracket$.
- Closed PCF terms $M : \tau \mapsto$ elements $\llbracket M \rrbracket \in \llbracket \tau \rrbracket$.
Denotations of open terms will be continuous functions.
- **Compositionality**.
In particular: $\llbracket M \rrbracket = \llbracket M' \rrbracket \Rightarrow \llbracket \mathcal{C}[M] \rrbracket = \llbracket \mathcal{C}[M'] \rrbracket$.
- **Soundness**.
For any type τ , $M \Downarrow_{\tau} V \Rightarrow \llbracket M \rrbracket = \llbracket V \rrbracket$.
- **Adequacy**.
For $\tau = \mathit{bool}$ or nat , $\llbracket M \rrbracket = \llbracket V \rrbracket \in \llbracket \tau \rrbracket \implies M \Downarrow_{\tau} V$.

Theorem. For all types τ 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 $M_1 \cong_{\text{ctx}} M_2 : \tau$.

Theorem. For all types τ 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 $M_1 \cong_{\text{ctx}} M_2 : \tau$.

Proof.

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

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

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

and symmetrically. □

Proof principle

To prove

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

it suffices to establish

$$\llbracket M_1 \rrbracket = \llbracket M_2 \rrbracket \text{ in } \llbracket \tau \rrbracket$$

Proof principle

To prove

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

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

$$\llbracket \Gamma \vdash M \rrbracket : \llbracket \Gamma \rrbracket \rightarrow \llbracket \tau \rrbracket$$

between domains.

Denotational semantics of PCF types

$\llbracket nat \rrbracket \stackrel{\text{def}}{=} \mathbb{N}_\perp$ (flat domain)

$\llbracket bool \rrbracket \stackrel{\text{def}}{=} \mathbb{B}_\perp$ (flat domain)

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

Denotational semantics of PCF types

$\llbracket nat \rrbracket \stackrel{\text{def}}{=} \mathbb{N}_\perp$ (flat domain)

$\llbracket bool \rrbracket \stackrel{\text{def}}{=} \mathbb{B}_\perp$ (flat domain)

$\llbracket \tau \rightarrow \tau' \rrbracket \stackrel{\text{def}}{=} \llbracket \tau \rrbracket \rightarrow \llbracket \tau' \rrbracket$ (function domain).

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

Denotational semantics of PCF type environments

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

Denotational semantics of PCF type environments

$$\begin{aligned} \llbracket \Gamma \rrbracket &\stackrel{\text{def}}{=} \prod_{x \in \text{dom}(\Gamma)} \llbracket \Gamma(x) \rrbracket \quad (\Gamma\text{-environments}) \\ &= \text{the domain of partial functions } \rho \text{ from variables} \\ &\text{to domains such that } \text{dom}(\rho) = \text{dom}(\Gamma) \text{ and} \\ &\rho(x) \in \llbracket \Gamma(x) \rrbracket \text{ for all } x \in \text{dom}(\Gamma) \end{aligned}$$

Denotational semantics of PCF type environments

$$\begin{aligned} \llbracket \Gamma \rrbracket &\stackrel{\text{def}}{=} \prod_{x \in \text{dom}(\Gamma)} \llbracket \Gamma(x) \rrbracket \quad (\Gamma\text{-environments}) \\ &= \text{the domain of partial functions } \rho \text{ from variables} \\ &\text{to domains such that } \text{dom}(\rho) = \text{dom}(\Gamma) \text{ and} \\ &\rho(x) \in \llbracket \Gamma(x) \rrbracket \text{ for all } x \in \text{dom}(\Gamma) \end{aligned}$$

Example:

1. For the empty type environment \emptyset ,

$$\llbracket \emptyset \rrbracket = \{ \perp \}$$

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

$$2. \llbracket \langle x \mapsto \tau \rangle \rrbracket = (\{ x \} \rightarrow \llbracket \tau \rrbracket)$$

$$2. \llbracket \langle x \mapsto \tau \rangle \rrbracket = (\{x\} \rightarrow \llbracket \tau \rrbracket) \cong \llbracket \tau \rrbracket$$

$$2. \llbracket \langle x \mapsto \tau \rangle \rrbracket = (\{x\} \rightarrow \llbracket \tau \rrbracket) \cong \llbracket \tau \rrbracket$$

3.

$$\begin{aligned} & \llbracket \langle x_1 \mapsto \tau_1, \dots, x_n \mapsto \tau_n \rangle \rrbracket \\ & \cong (\{x_1\} \rightarrow \llbracket \tau_1 \rrbracket) \times \dots \times (\{x_n\} \rightarrow \llbracket \tau_n \rrbracket) \\ & \cong \llbracket \tau_1 \rrbracket \times \dots \times \llbracket \tau_n \rrbracket \end{aligned}$$

Denotational semantics of PCF terms, I

$$\llbracket \Gamma \vdash \mathbf{0} \rrbracket (\rho) \stackrel{\text{def}}{=} 0 \in \llbracket \mathit{nat} \rrbracket$$

$$\llbracket \Gamma \vdash \mathbf{true} \rrbracket (\rho) \stackrel{\text{def}}{=} \mathit{true} \in \llbracket \mathit{bool} \rrbracket$$

$$\llbracket \Gamma \vdash \mathbf{false} \rrbracket (\rho) \stackrel{\text{def}}{=} \mathit{false} \in \llbracket \mathit{bool} \rrbracket$$

Denotational semantics of PCF terms, I

$$\llbracket \Gamma \vdash \mathbf{0} \rrbracket (\rho) \stackrel{\text{def}}{=} 0 \in \llbracket \mathit{nat} \rrbracket$$

$$\llbracket \Gamma \vdash \mathbf{true} \rrbracket (\rho) \stackrel{\text{def}}{=} \mathit{true} \in \llbracket \mathit{bool} \rrbracket$$

$$\llbracket \Gamma \vdash \mathbf{false} \rrbracket (\rho) \stackrel{\text{def}}{=} \mathit{false} \in \llbracket \mathit{bool} \rrbracket$$

$$\llbracket \Gamma \vdash x \rrbracket (\rho) \stackrel{\text{def}}{=} \rho(x) \in \llbracket \Gamma(x) \rrbracket \quad (x \in \mathit{dom}(\Gamma))$$

Denotational semantics of PCF terms, II

$\llbracket \Gamma \vdash \mathbf{succ}(M) \rrbracket(\rho)$

$$\underline{\underline{\text{def}}} \begin{cases} \llbracket \Gamma \vdash M \rrbracket(\rho) + 1 & \text{if } \llbracket \Gamma \vdash M \rrbracket(\rho) \neq \perp \\ \perp & \text{if } \llbracket \Gamma \vdash M \rrbracket(\rho) = \perp \end{cases}$$

Denotational semantics of PCF terms, II

$\llbracket \Gamma \vdash \mathbf{succ}(M) \rrbracket(\rho)$

$$\stackrel{\text{def}}{=} \begin{cases} \llbracket \Gamma \vdash M \rrbracket(\rho) + 1 & \text{if } \llbracket \Gamma \vdash M \rrbracket(\rho) \neq \perp \\ \perp & \text{if } \llbracket \Gamma \vdash M \rrbracket(\rho) = \perp \end{cases}$$

$\llbracket \Gamma \vdash \mathbf{pred}(M) \rrbracket(\rho)$

$$\stackrel{\text{def}}{=} \begin{cases} \llbracket \Gamma \vdash M \rrbracket(\rho) - 1 & \text{if } \llbracket \Gamma \vdash M \rrbracket(\rho) > 0 \\ \perp & \text{if } \llbracket \Gamma \vdash M \rrbracket(\rho) = 0, \perp \end{cases}$$

Denotational semantics of PCF terms, II

$\llbracket \Gamma \vdash \mathbf{succ}(M) \rrbracket(\rho)$

$$\stackrel{\text{def}}{=} \begin{cases} \llbracket \Gamma \vdash M \rrbracket(\rho) + 1 & \text{if } \llbracket \Gamma \vdash M \rrbracket(\rho) \neq \perp \\ \perp & \text{if } \llbracket \Gamma \vdash M \rrbracket(\rho) = \perp \end{cases}$$

$\llbracket \Gamma \vdash \mathbf{pred}(M) \rrbracket(\rho)$

$$\stackrel{\text{def}}{=} \begin{cases} \llbracket \Gamma \vdash M \rrbracket(\rho) - 1 & \text{if } \llbracket \Gamma \vdash M \rrbracket(\rho) > 0 \\ \perp & \text{if } \llbracket \Gamma \vdash M \rrbracket(\rho) = 0, \perp \end{cases}$$

$$\llbracket \Gamma \vdash \mathbf{zero}(M) \rrbracket(\rho) \stackrel{\text{def}}{=} \begin{cases} \mathit{true} & \text{if } \llbracket \Gamma \vdash M \rrbracket(\rho) = 0 \\ \mathit{false} & \text{if } \llbracket \Gamma \vdash M \rrbracket(\rho) > 0 \\ \perp & \text{if } \llbracket \Gamma \vdash M \rrbracket(\rho) = \perp \end{cases}$$

Denotational semantics of PCF terms, III

$\llbracket \Gamma \vdash \mathbf{if} \ M_1 \ \mathbf{then} \ M_2 \ \mathbf{else} \ M_3 \rrbracket (\rho)$

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

Denotational semantics of PCF terms, III

$\llbracket \Gamma \vdash \text{if } M_1 \text{ then } M_2 \text{ else } M_3 \rrbracket(\rho)$

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

$$\llbracket \Gamma \vdash M_1 M_2 \rrbracket(\rho) \stackrel{\text{def}}{=} (\llbracket \Gamma \vdash M_1 \rrbracket(\rho)) (\llbracket \Gamma \vdash M_2 \rrbracket(\rho))$$

Denotational semantics of PCF terms, IV

$$\begin{aligned} & \llbracket \Gamma \vdash \mathbf{fn} \ x : \tau . M \rrbracket (\rho) \\ & \stackrel{\text{def}}{=} \lambda d \in \llbracket \tau \rrbracket . \llbracket \Gamma[x \mapsto \tau] \vdash M \rrbracket (\rho[x \mapsto d]) \quad (x \notin \text{dom}(\Gamma)) \end{aligned}$$

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

$$\llbracket \Gamma \vdash \mathbf{fix}(M) \rrbracket(\rho) \stackrel{\text{def}}{=} \mathit{fix}(\llbracket \Gamma \vdash M \rrbracket(\rho))$$

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*

$$\llbracket \Gamma \vdash M \rrbracket : \llbracket \Gamma \rrbracket \rightarrow \llbracket \tau \rrbracket$$

is a well-defined continuous function.

Denotations of closed terms

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

$$\llbracket \emptyset \vdash M \rrbracket : \llbracket \emptyset \rrbracket \rightarrow \llbracket \tau \rrbracket$$

and, since $\llbracket \emptyset \rrbracket = \{ \perp \}$, we have

$$\llbracket M \rrbracket \stackrel{\text{def}}{=} \llbracket \emptyset \vdash M \rrbracket (\perp) \in \llbracket \tau \rrbracket \quad (M \in \text{PCF}_\tau)$$

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 $\Gamma' \vdash \mathcal{C}[M'] : \tau'$,

if $[[\Gamma \vdash M]] = [[\Gamma \vdash M']] : [[\Gamma]] \rightarrow [[\tau]]$

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

Soundness

Proposition. *For all closed terms $M, V \in \text{PCF}_\tau$,*
if $M \Downarrow_\tau V$ then $\llbracket M \rrbracket = \llbracket V \rrbracket \in \llbracket \tau \rrbracket$.

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,

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

for all $\rho \in \llbracket \Gamma \rrbracket$.

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,

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

for all $\rho \in \llbracket \Gamma \rrbracket$.

In particular when $\Gamma = \emptyset$, $\llbracket \langle x \mapsto \tau \rangle \vdash M' \rrbracket : \llbracket \tau \rrbracket \rightarrow \llbracket \tau' \rrbracket$ and

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

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

$$\llbracket M \rrbracket = \llbracket V \rrbracket \in \llbracket \gamma \rrbracket \implies M \Downarrow_{\gamma} V .$$

Adequacy

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

$$\llbracket M \rrbracket = \llbracket V \rrbracket \in \llbracket \gamma \rrbracket \implies M \Downarrow_{\gamma} V .$$

NB. Adequacy does not hold at function types

Adequacy

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

$$\llbracket M \rrbracket = \llbracket V \rrbracket \in \llbracket \gamma \rrbracket \implies M \Downarrow_{\gamma} V.$$

NB. Adequacy does not hold at function types:

$$\llbracket \mathbf{fn} \ x : \tau. (\mathbf{fn} \ y : \tau. y) \ x \rrbracket \quad = \quad \llbracket \mathbf{fn} \ x : \tau. x \rrbracket \quad : \llbracket \tau \rrbracket \rightarrow \llbracket \tau \rrbracket$$

Adequacy

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

$$\llbracket M \rrbracket = \llbracket V \rrbracket \in \llbracket \gamma \rrbracket \implies M \Downarrow_{\gamma} V.$$

NB. Adequacy does not hold at function types:

$$\llbracket \mathbf{fn} \ x : \tau. (\mathbf{fn} \ y : \tau. y) \ x \rrbracket = \llbracket \mathbf{fn} \ x : \tau. x \rrbracket : \llbracket \tau \rrbracket \rightarrow \llbracket \tau \rrbracket$$

but

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

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:

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

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 $\gamma \in \{nat, bool\}$,

$$\llbracket M \rrbracket \triangleleft_{\gamma} M \text{ implies } \underbrace{\forall V (\llbracket M \rrbracket = \llbracket V \rrbracket \implies M \downarrow_{\gamma} V)}_{\text{adequacy}}$$

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

$$n \triangleleft_{\text{nat}} M \stackrel{\text{def}}{\iff} (n \in \mathbb{N} \Rightarrow M \Downarrow_{\text{nat}} \mathbf{succ}^n(\mathbf{0}))$$

$$b \triangleleft_{\text{bool}} M \stackrel{\text{def}}{\iff} (b = \text{true} \Rightarrow M \Downarrow_{\text{bool}} \mathbf{true}) \\ \& (b = \text{false} \Rightarrow M \Downarrow_{\text{bool}} \mathbf{false})$$

Proof of: $\llbracket M \rrbracket \triangleleft_\gamma M$ implies adequacy

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

$$\llbracket M \rrbracket = \llbracket V \rrbracket$$

$$\implies \llbracket M \rrbracket = \llbracket \text{succ}^n(\mathbf{0}) \rrbracket \quad \text{for some } n \in \mathbb{N}$$

$$\implies n = \llbracket M \rrbracket \triangleleft_\gamma M$$

$$\implies M \Downarrow \text{succ}^n(\mathbf{0}) \quad \text{by definition of } \triangleleft_{\text{nat}}$$

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 $M = M_1 M_2$.

\rightsquigarrow *logical* definition

Definition of

$$f \triangleleft_{\tau \rightarrow \tau'} M \quad (f \in (\llbracket \tau \rrbracket \rightarrow \llbracket \tau' \rrbracket), M \in \text{PCF}_{\tau \rightarrow \tau'})$$

Definition of

$$f \triangleleft_{\tau \rightarrow \tau'} M \quad (f \in (\llbracket \tau \rrbracket \rightarrow \llbracket \tau' \rrbracket), M \in \text{PCF}_{\tau \rightarrow \tau'})$$

$$f \triangleleft_{\tau \rightarrow \tau'} M$$

$$\stackrel{\text{def}}{\Leftrightarrow} \forall x \in \llbracket \tau \rrbracket, N \in \text{PCF}_{\tau}$$

$$(x \triangleleft_{\tau} N \Rightarrow f(x) \triangleleft_{\tau'} M N)$$

Requirements on the formal approximation relations, III

We want to be able to proceed by induction.

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

\rightsquigarrow *admissibility* property

Admissibility property

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

$$\{ d \in \llbracket \tau \rrbracket \mid d \triangleleft_\tau M \}$$

is an admissible subset of $\llbracket \tau \rrbracket$.

Further properties

Lemma. For all types τ , elements $d, d' \in \llbracket \tau \rrbracket$, and terms $M, N, V \in \text{PCF}_\tau$,

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

Requirements on the formal approximation relations, IV

We want to be able to proceed by induction.

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

\rightsquigarrow *substitutivity* property for open terms

Fundamental property

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

Fundamental property

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

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 \triangleleft_{τ}

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

$$\rho \triangleleft_{\Gamma} \sigma \Rightarrow \llbracket \Gamma \vdash M \rrbracket(\rho) \triangleleft_{\tau} M[\sigma]$$

-
- $\rho \triangleleft_{\Gamma} \sigma$ means that $\rho(x) \triangleleft_{\Gamma(x)} \sigma(x)$ 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 τ , and a type environment Γ , the relation $\Gamma \vdash M_1 \leq_{\text{ctx}} M_2 : \tau$ is defined to hold iff

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

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

Extensionality properties of \leq_{ctx}

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

$M_1 \leq_{\text{ctx}} M_2 : \gamma$ 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 $\tau \rightarrow \tau'$,

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

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

Topic 8

Full Abstraction

Proof principle

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

$$\llbracket M_1 \rrbracket = \llbracket M_2 \rrbracket \text{ in } \llbracket \tau \rrbracket \implies M_1 \cong_{\text{ctx}} M_2 : \tau .$$

Hence, to prove

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

it suffices to establish

$$\llbracket M_1 \rrbracket = \llbracket M_2 \rrbracket \text{ in } \llbracket \tau \rrbracket .$$

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

$$T_1, T_2 \in \text{PCF}_{(bool \rightarrow (bool \rightarrow bool)) \rightarrow bool}$$

such that

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

and

$$\llbracket T_1 \rrbracket \neq \llbracket T_2 \rrbracket$$

► We achieve $T_1 \cong_{\text{ctx}} T_2$ by making sure that

$$\forall M \in \text{PCF}_{\text{bool} \rightarrow (\text{bool} \rightarrow \text{bool})} (T_1 M \Downarrow_{\text{bool}} \& T_2 M \Downarrow_{\text{bool}})$$

► We achieve $T_1 \cong_{\text{ctx}} T_2$ by making sure that

$$\forall M \in \text{PCF}_{\text{bool} \rightarrow (\text{bool} \rightarrow \text{bool})} (T_1 M \Downarrow_{\text{bool}} \ \& \ T_2 M \Downarrow_{\text{bool}})$$

Hence,

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

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

- We achieve $T_1 \cong_{\text{ctx}} T_2$ 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,

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

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

- 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

$$\text{por} \in (\mathbb{B}_\perp \rightarrow (\mathbb{B}_\perp \rightarrow \mathbb{B}_\perp)) .$$

Parallel-or function

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

$$por \ true \ \perp \quad = \ true$$

$$por \ \perp \ true \quad = \ true$$

$$por \ false \ false \quad = \ false$$

Parallel-or function

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

$$por \ true \ \perp \quad = \ true$$

$$por \ \perp \ true \quad = \ true$$

$$por \ false \ false \quad = \ false$$

In which case, it necessarily follows by monotonicity that

$$por \ true \ true \quad = \ true \qquad por \ false \ \perp \quad = \ \perp$$

$$por \ true \ false \quad = \ true \qquad por \ \perp \ false \quad = \ \perp$$

$$por \ false \ true \quad = \ true \qquad por \ \perp \ \perp \quad = \ \perp$$

Undefinability of parallel-or

Proposition. *There is no closed PCF term*

$$P : \text{bool} \rightarrow (\text{bool} \rightarrow \text{bool})$$

satisfying

$$\llbracket P \rrbracket = \text{por} : \mathbb{B}_\perp \rightarrow (\mathbb{B}_\perp \rightarrow \mathbb{B}_\perp) .$$

Parallel-or test functions

Parallel-or test functions

For $i = 1, 2$ define

$$T_i \stackrel{\text{def}}{=} \text{fn } f : \text{bool} \rightarrow (\text{bool} \rightarrow \text{bool}) .$$
$$\quad \text{if } (f \text{ true } \Omega) \text{ then}$$
$$\quad \quad \text{if } (f \ \Omega \ \text{true}) \text{ then}$$
$$\quad \quad \quad \text{if } (f \ \text{false} \ \text{false}) \text{ then } \Omega \ \text{else } B_i$$
$$\quad \quad \quad \text{else } \Omega$$
$$\quad \text{else } \Omega$$

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

Failure of full abstraction

Proposition.

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

$$\llbracket T_1 \rrbracket \neq \llbracket T_2 \rrbracket \in (\mathbb{B}_\perp \rightarrow (\mathbb{B}_\perp \rightarrow \mathbb{B}_\perp)) \rightarrow \mathbb{B}_\perp$$

PCF+por

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

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

Evaluation

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

Plotkin's full abstraction result

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

$$\llbracket \Gamma \vdash \mathbf{por}(M_1, M_2) \rrbracket(\rho) \stackrel{\text{def}}{=} \mathit{por}(\llbracket \Gamma \vdash M_1 \rrbracket(\rho))(\llbracket \Gamma \vdash M_2 \rrbracket(\rho))$$

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

$$\Gamma \vdash M_1 \cong_{\text{ctx}} M_2 : \tau \iff \llbracket \Gamma \vdash M_1 \rrbracket = \llbracket \Gamma \vdash M_2 \rrbracket.$$