Where we're at

We have a denotational semantics for types $[\![\tau]\!]$ and terms $[\![t]\!]$ such that:

Compositionality: $[\![t]\!] = [\![t']\!] \Rightarrow [\![\mathcal{C}[t]]\!] = [\![\mathcal{C}[t']]\!]$. ✓

Soundness: for any type τ, $t \Downarrow_\tau v \Rightarrow [\![t]\!] = [\![v]\!]$. ✓

Adequacy: for $\gamma = \texttt{bool}$ or $\texttt{nat}$, if $t \in \mathrm{PCF}_\gamma$ and $[\![t]\!] = [\![v]\!]$ then $t \Downarrow_\gamma v$. ✓

Where we're at

We have a denotational semantics for types $[\![\tau]\!]$ and terms $[\![t]\!]$ such that:

Compositionality: $[\![t]\!] = [\![t']\!] \Rightarrow [\![\mathcal{C}[t]]\!] = [\![\mathcal{C}[t']]\!]$. ✓

Soundness: for any type τ, $t \Downarrow_\tau v \Rightarrow [\![t]\!] = [\![v]\!]$. ✓

Adequacy: for $\gamma = \texttt{bool}$ or $\texttt{nat}$, if $t \in \mathrm{PCF}_\gamma$ and $[\![t]\!] = [\![v]\!]$ then $t \Downarrow_\gamma v$. ✓

From this we can show

$$[\![t]\!] = [\![u]\!] \in [\![\tau]\!] \Rightarrow t \cong_{\mathrm{ctx}} u : \tau$$

What about the converse implication?

Full abstraction

Full abstraction

Failure of full abstraction

Full abstraction

A denotational model is **fully abstract** if

$$t_1 \cong_{\mathrm{ctx}} t_2 : \tau \Rightarrow [\![t_1]\!] = [\![t_2]\!] \in [\![\tau]\!]$$

Full abstraction

A denotational model is fully abstract if

$$t_1 \cong_{\text{ctx}} t_2 : \tau \Rightarrow [\![t_1]\!] = [\![t_2]\!] \in [\![\tau]\!]$$

A form of completeness of semantic equivalence wrt. program equivalence.

Full abstraction

A denotational model is **fully abstract** if

$$t_1 \cong_{\mathrm{ctx}} t_2 : \tau \Rightarrow [\![t_1]\!] = [\![t_2]\!] \in [\![\tau]\!]$$

A form of **completeness** of semantic equivalence wrt. program equivalence.

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

Parallel or

The *parallel or* function $\mathbf{por} : \mathbb{B}_\perp \times \mathbb{B}_\perp \to \mathbb{B}_\perp$ is defined as given by the following table:

por	true	false	$\perp$
true	true	true	true
false	true	false	$\perp$
$\perp$	true	$\perp$	$\perp$

The (left) sequential or function $\mathbf{or} : \mathbb{B}_\perp \times \mathbb{B}_\perp \to \mathbb{B}_\perp$ is defined as

$$\mathrm{or} \stackrel{\mathrm{def}}{=} [\![\texttt{fun}\ x\colon \texttt{bool}.\ \texttt{fun}\ y\colon \texttt{bool}.\ \texttt{if}\ x\ \texttt{then}\ \texttt{true}\ \texttt{else}\ y]\!]$$

It is given by the following table:

or	true	false	$\perp$
true	true	true	true
false	true	false	$\perp$
$\perp$	$\perp$	$\perp$	$\perp$

Parallel vs sequential or

por	true	false	⊥
true	true	true	true
false	true	false	⊥
⊥	true	⊥	⊥

or	true	false	⊥
true	true	true	true
false	true	false	⊥
⊥	⊥	⊥	⊥

Parallel vs sequential or

por	true	false	⊥
true	true	true	true
false	true	false	⊥
⊥	true	⊥	⊥

or	true	false	⊥
true	true	true	true
false	true	false	⊥
⊥	⊥	⊥	⊥

or is sequential, but por is not.

Undefinability or parallel or

There is no closed PCF term

$$t : \texttt{bool} \to \texttt{bool} \to \texttt{bool}$$

satisfying

$$[\![t]\!] = \mathrm{por} : \mathbb{B}_\perp \to \mathbb{B}_\perp \to \mathbb{B}_\perp \ .$$

Failure of full abstraction

The denotational model of PCF in domains and continuous functions is not fully abstract.

The denotational model of PCF in domains and continuous functions is not fully abstract.

For well-chosen $T_{\texttt{true}}$ and $T_{\texttt{false}}$,

$$T_{\texttt{true}} \cong_{\text{ctx}} T_{\texttt{false}} : (\texttt{bool} \to \texttt{bool} \to \texttt{bool}) \to \texttt{bool}$$

$$[\![T_{\texttt{true}}]\!] \neq [\![T_{\texttt{false}}]\!] \in (\mathbb{B} \to \mathbb{B} \to \mathbb{B}) \to \mathbb{B}$$

The denotational model of PCF in domains and continuous functions is not fully abstract.

For well-chosen $T_{\texttt{true}}$ and $T_{\texttt{false}}$,

$$T_{\texttt{true}} \cong_{\text{ctx}} T_{\texttt{false}} : (\texttt{bool} \to \texttt{bool} \to \texttt{bool}) \to \texttt{bool}$$

$$[\![T_{\texttt{true}}]\!] \neq [\![T_{\texttt{false}}]\!] \in (\mathbb{B} \to \mathbb{B} \to \mathbb{B}) \to \mathbb{B}$$

Idea:

- for all $f \in PCF_{\texttt{bool}\to\texttt{bool}\to\texttt{bool}}$, ensure $T_b\ f \Uparrow_{\texttt{bool}}$...
- but $[\![T_b]\!]\ (\text{por}) = [\![b]\!]$.

$$
\begin{aligned}
T_b \stackrel{\text{def}}{=}\ & \texttt{fun}\ f\colon \texttt{bool} \to (\texttt{bool} \to \texttt{bool}).\\
& \quad \texttt{if}(f\ \texttt{true}\ \Omega_{\texttt{bool}})\ \texttt{then}\\
& \qquad \texttt{if}\ (f\ \Omega_{\texttt{bool}}\ \texttt{true})\ \texttt{then}\\
& \qquad\quad \texttt{if}\ (f\ \texttt{false}\ \texttt{false})\ \texttt{then}\ \Omega_{\texttt{bool}}\ \texttt{else}\ b\\
& \qquad \texttt{else}\ \Omega_{\texttt{bool}}\\
& \quad \texttt{else}\ \Omega_{\texttt{bool}}
\end{aligned}
$$

1) $T_b f \Uparrow_{bool}$ for all $f \in PCF_{bool \to bool \to bool}$

$T_b f \Downarrow_{bool} v$ iff

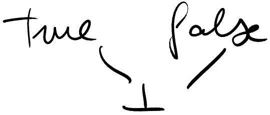
$$\left.\begin{array}{l} f \ \mathsf{true}\ \Omega_{bool} \Downarrow \mathsf{true} \\ f\ \Omega_{bool}\ \mathsf{true} \Downarrow \mathsf{true} \\ f\ \mathsf{false}\ \mathsf{false} \Downarrow \mathsf{false} \end{array}\right\} (1)$$

if f satisfies (1) then

$$\begin{array}{l} [\![f]\!](\mathsf{true}, \bot_B) = \mathsf{true} \\ [\![f]\!](\bot_B, \mathsf{true}) = \mathsf{true} \\ [\![f]\!](\mathsf{false}, \mathsf{false}) = \mathsf{false} \end{array} \quad (2)$$

⟦f⟧	true	false	⊥
true	true	true	true
false	true	false	⊥
⊥	true	⊥	⊥

(e) ⇒ ⟦f⟧ = por



∄ cannot exist

so for every $f \in PCF_{bool \to bool \to bool}$ $[\![f]\!]_{bool}$

a) Dut $[\![\neg b]\!](\rho\sigma) = \neg [\![b]\!]$

Full abstraction

Beyond full abstraction failure

Interpreting full abstraction failure

- PCF is not expressive enough to present the model?
- The model does not adequately capture PCF?
- Contexts are too weak: they do not distinguish enough programs?

PCF+por

$\boxed{\Gamma \vdash t : \tau}$

$$\cdots \qquad \text{POR}\ \frac{\Gamma \vdash t_1 : \tau \qquad \Gamma \vdash t_2 : \tau}{\Gamma \vdash \mathtt{por}(t_1, t_2) : \tau}$$

$\boxed{t \Downarrow_\tau v}$

$$\text{PORL}\ \frac{t_1 \Downarrow_{\mathtt{bool}} \mathtt{true}}{\mathtt{por}(t_1, t_2) \Downarrow_{\mathtt{bool}} \mathtt{true}} \qquad \text{PORR}\ \frac{t_2 \Downarrow_{\mathtt{bool}} \mathtt{true}}{\mathtt{por}(t_1, t_2) \Downarrow_{\mathtt{bool}} \mathtt{true}}$$

$$\text{PORF}\ \frac{t_1 \Downarrow_{\mathtt{bool}} \mathtt{false} \qquad t_2 \Downarrow_{\mathtt{bool}} \mathtt{false}}{\mathtt{por}(t_1, t_2) \Downarrow_{\mathtt{bool}} \mathtt{false}}$$

Full abstraction for PCF+`por`

If we extend the semantics of PCF to PCF+`por` with

$$[\![\texttt{por}]\!] = \mathrm{por}$$

the resulting denotational semantics is fully abstract.

If we extend the semantics of PCF to PCF+`por` with

$$[\![\texttt{por}]\!] = \mathrm{por}$$

the resulting denotational semantics is fully abstract…

but is PCF+`por` still a reasonable model of programming language?

Fully abstract semantics for PCF

- first step: dI-domains & stable functions → no $\mathbf{por}$ any more, but still not fully abstract…
- only proper answers in the late 90s (!): logical relations and game semantics

Fully abstract semantics

Fully abstract semantics for PCF

- first step: dI-domains & stable functions $\rightarrow$ no $\mathbf{por}$ any more, but still not fully abstract…
- only proper answers in the late 90s (!): logical relations and game semantics

Real languages have effects

- If you add effects (references, control flow…) to a language, contexts become *much more* expressive.
- Full abstraction becomes different: somewhat easier… but is contextual equivalence still a reasonable idea?

Where to go from here?

Source of a very rich literature:

- linear logic
- logical relations
- game semantics
- bisimulations techniques
- ...

Separate

- the structure needed to interpret a language (generic)
- how to construct this structure in particular examples (specific)

Separate

- the structure needed to interpret a language (generic)
- how to construct this structure in particular examples (specific)

Interpret:

- a type τ as an object in a category;
- a term $\Gamma \vdash t : \tau$ as a morphism/arrow $[\![t]\!] : [\![\Gamma]\!] \to [\![\tau]\!]$.

Categorical semantics

Separate

- the structure needed to interpret a language (generic)
- how to construct this structure in particular examples (specific)

Interpret:

- a type τ as an object in a category;
- a term $\Gamma \vdash t : \tau$ as a morphism/arrow $[\![t]\!] : [\![\Gamma]\!] \to [\![\tau]\!]$.

Example: λ-calculus → cartesian closed categories

OCaml's ADT:

```
type 'a tree =
  | Leaf
  | Node of 'a * 'a tree * 'a tree
```

It is a fixed point equation! We can use domain theory to solve it.

Beyond pure languages

Effects: control flow (errors), mutability/state, input-output...
An important aspect of programming languages!

Effects: control flow (errors), mutability/state, input-output...
An important aspect of programming languages!

Modelled as a **monad** T (example: $T(A) \stackrel{\text{def}}{=} (A \times \text{State})^{\text{State}}$)

Effects: control flow (errors), mutability/state, input-output...
An important aspect of programming languages!

Modelled as a monad T (example: $T(A) \overset{\text{def}}{=} (A \times \text{State})^{\text{State}}$)

$[\![\Gamma]\!] \to ([\![\tau]\!] \times \text{State})^{\text{State}}$

$\cong$

$[\![\Gamma]\!] \times \text{State} \to [\![\tau]\!] \times \text{State}$

Denotation of a computation: $[\![\Gamma]\!] \to T([\![\tau]\!])$

$[\![\Gamma]\!] \to T([\![\sigma]\!])$

More semantics

Easter: **axiomatic semantic** (Hoare Logic and Model Checking)

More semantics

Easter: axiomatic semantic (Hoare Logic and Model Checking)

In the end, the most interesting aspects of semantics is in the interaction between different approaches.