Contextual Equivalence

[§ 5.5 , p 44]

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.

untyped λ-calculus equivalent is YM
where $Y = \lambda f.(\lambda x. f(xx))(\lambda x. f(xx))$
(Y is not typeable with PCF's simple types)

Contextual equivalence

Two phrases of a programming language are ("Morris style") contextually equivalent ($\cong_{\mathrm{ctx}}$) if occurrences of the first phrase in any program can be replaced by the second phrase without affecting the observable results of executing the program.

We assume the programming language comes with an operational semantics as part of its definition

E.g. PCF term for addition

$$
\begin{aligned}
&\mathrm{fix}\,(\mathrm{fn}\ p : nat \to nat \to nat.\ \mathrm{fn}\ x : nat.\ \mathrm{fn}\ y : nat\\
&\quad\quad \mathrm{if}\ \mathrm{zero}(y)\ \mathrm{then}\ x\\
&\quad\quad \mathrm{else}\ \mathrm{succ}(p\,x\,(\mathrm{pred}(y))))
\end{aligned}
$$

E.g. PCF term for addition

$$\mathrm{fix}\,(\mathrm{fn}\ p : nat \to nat \to nat.\ \mathrm{fn}\ x : nat.\ \mathrm{fn}\ y : nat$$
$$\quad \mathrm{if}\ \mathrm{zero}(y)\ \mathrm{then}\ \mathrm{pred}(\mathrm{succ}(x))$$
$$\quad \mathrm{else}\ \mathrm{succ}(p\,x\,(\mathrm{pred}(y)))\,)$$

expect that x and $\mathrm{pred}(\mathrm{succ}\,x)$ are contextually equivalent for PCF

PCF contexts $\mathcal{C}$

$$\mathcal{C} ::= \_ \mid 0 \mid \mathrm{succ}(\mathcal{C}) \mid \mathrm{pred}(\mathcal{C})$$
$$\mid \mathrm{zero}(\mathcal{C}) \mid \mathrm{true} \mid \mathrm{false} \mid \mathrm{if}\ \mathcal{C}\ \mathrm{then}\ \mathcal{C}\ \mathrm{else}\ \mathcal{C}$$
$$\mid x \mid \mathrm{fn}\, x : \tau .\, \mathcal{C} \mid \mathcal{C}\,\mathcal{C} \mid \mathrm{fix}(\mathcal{C})$$

a "hole", or place holder, to be filled with a PCF term

PCF contexts $\mathcal{C}$

$$
\begin{aligned}
\mathcal{C} ::= &\ - \mid 0 \mid \mathrm{succ}(\mathcal{C}) \mid \mathrm{pred}(\mathcal{C}) \\
&\mid \mathrm{zero}(\mathcal{C}) \mid \mathrm{true} \mid \mathrm{false} \mid \mathrm{if}\ \mathcal{C}\ \mathrm{then}\ \mathcal{C}\ \mathrm{else}\ \mathcal{C} \\
&\mid x \mid \mathrm{fn}\, x : \tau .\, \mathcal{C} \mid \mathcal{C}\ \mathcal{C} \mid \mathrm{fix}(\mathcal{C})
\end{aligned}
$$

Notation: $\mathcal{C}[M]$ = PCF term obtained from $\mathcal{C}$ by replacing all occurrences of $-$ by M

$$\mathrm{fix}\,(\mathrm{fn}\ p : nat \to nat \to nat.\ \mathrm{fn}\ x : nat.\ \mathrm{fn}\ y : nat$$
$$\quad \mathrm{if}\ \mathrm{zero}(y)\ \mathrm{then}\ \mathrm{pred}(\mathrm{succ}(x))$$
$$\quad \mathrm{else}\ \mathrm{succ}(p\,x\,(\mathrm{pred}(y))))$$

is $\mathcal{C}[\mathrm{pred}(\mathrm{succ}(x))]$ for

$$\mathcal{C} = \mathrm{fix}\,(\mathrm{fn}\ p : nat \to nat \to nat.\ \mathrm{fn}\ x : nat.\ \mathrm{fn}\ y : nat$$
$$\quad \mathrm{if}\ \mathrm{zero}(y)\ \mathrm{then}\ -$$
$$\quad \mathrm{else}\ \mathrm{succ}(p\,x\,(\mathrm{pred}(y))))$$

Contextual equivalences

Two phrases of a programming language are ("Morris style") contextually equivalent ($\cong_{\mathrm{ctx}}$) if occurrences of the first phrase in any program can be replaced by the second phrase without affecting the observable results of executing the program.

Different choices lead to possibly different notions of contextual equivalence.

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 $V : \gamma$,

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

When $\Gamma = \emptyset$, just write $\emptyset \vdash M_1 \cong_{ctx} M_2 : \tau$ as

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

Examples of PCF contextual equivalence

$\{x : nat\} \vdash pred(succ(x)) \cong_{ctx} x : nat$

$\{x : nat\} \vdash zero(0) \cong_{ctx} true : bool$

? $\{x : nat\} \vdash zero(succ(x)) \cong_{ctx} false : bool$

~~Non~~ Examples of PCF contextual equivalence

$\{x : nat\} \vdash pred(succ(x)) \cong_{ctx} x : nat$

$\{x : nat\} \vdash zero(0) \cong_{ctx} true : bool$

$\{x : nat\} \vdash zero(succ(x)) \not\cong_{ctx} false : bool$

because for $\mathcal{C} = (\lambda x : nat. -)\,\Omega_{nat}$ we have

$$\begin{cases} \mathcal{C}[zero(succ\,x)] = (\lambda x : nat.\, zero(succ\,x))\,\Omega_{nat} \not\Downarrow_{nat} \\ \mathcal{C}[false] = (\lambda x : nat.\, false)\,\Omega_{nat} \Downarrow_{nat} false \end{cases}$$

Non-Examples of PCF contextual equivalence

$\{x : nat\} \vdash pred(succ(x)) \cong_{ctx} x : nat$

$\{x : nat\} \vdash zero(0) \cong_{ctx} true : bool$

$\{x : nat\} \vdash zero(succ(x)) \not\cong_{ctx} false : bool$

because for $\mathcal{C} = (\lambda x : nat . -)\,\Omega_{nat}$ we have

$$\begin{cases} \mathcal{C}[zero(succ\, x)] = (\lambda x : nat .\, zero(succ\, x))\,\Omega_{nat} \not\Downarrow_{nat} \\ \mathcal{C}[false] = (\lambda x : nat .\, false)\,\Omega_{nat} \Downarrow_{nat} false \end{cases}$$

MORAL: easy to show $\not\cong_{ctx}$ (usually).

Examples of PCF contextual equivalence

$$(\lambda x : \tau .\, M)\, M' \cong_{ctx} M[M'/x] : \tau'$$

(where $\begin{cases} \lambda x : \tau .\, M : \tau \to \tau' \\ M' : \tau \end{cases}$)

$$M \cong_{ctx} \lambda x : \tau .\, M x : \tau \to \tau'$$

(where $M : \tau \to \tau'$
& $x \notin fv(M)$)

$$\mathbf{fix}(M) \cong_{ctx} M\, \mathbf{fix}(M) : \tau$$

(where $M : \tau \to \tau$)

HOW DOES ONE PROVE SUCH FACTS ?

PCF denotational semantics — aims

PCF denotational semantics — aims

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

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.

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.

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

$$\boxed{\begin{array}{l} [\![\Gamma]\!] = [\![\tau_1]\!] \times \cdots \times [\![\tau_n]\!] \\ \text{if } \Gamma = \{x_1 : \tau_1, \ldots, x_n : \tau_n\} \end{array}}$$

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

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

  not at function types, because...

Example 5.6.1 [p45]

$$\boxed{\begin{array}{l} V \triangleq \mathrm{fn}\ x : nat.\ (\mathrm{fn}\ y : nat.\ y)\ 0 \\ \hline V' \triangleq \mathrm{fn}\ x : nat.\ 0 \end{array}}$$

Satisfy: $V \not\Downarrow_{nat \to nat} V'$

because in general can only prove $V \Downarrow V'$ for $V' = V$

Example 5.6.1

$$V \triangleq \mathrm{fn}\ x : nat.\ (\mathrm{fn}\ y : nat.\ y)\ 0$$

$$V' \triangleq \mathrm{fn}\ x : nat.\ 0$$

Satisfy:

$$V \not\Downarrow_{nat \to nat} V'$$

$$[\![V]\!] = [\![V']\!]$$

because $(\mathrm{fn}\ y : nat.\ y)\ 0 \Downarrow_{nat} 0$

so $[\![(\mathrm{fn}\ y : nat\ y)\ 0]\!] = [\![0]\!]$ by Soundness

Example 5.6.1

$$\boxed{\begin{array}{l} V \triangleq \mathbf{fn}\ x : nat . (\mathbf{fn}\ y : nat . y)\, 0 \\ \hline V' \triangleq \mathbf{fn}\ x : nat .\ 0 \end{array}}$$

Satisfy: $V \not\Downarrow_{nat \to nat} V'$

$[\![V]\!] = [\![V']\!]$

because $(\mathbf{fn}\ y : nat . y)\, 0 \Downarrow_{nat} 0$

so $[\![(\mathbf{fn}\ y : nat . y)\, 0]\!] = [\![0]\!]$ by soundness

so $[\![\mathcal{C}[(\mathbf{fn}\ y : nat . y)\, 0]]\!] = [\![\mathcal{C}[0]]\!]$ by compositionality

and we can take $\mathcal{C} = \mathbf{fn}\ x : nat . -$.

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.

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

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

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

and symmetrically (& similarly for $\Downarrow_{bool}$). □

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?

Proof principle

To prove

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

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?

In Chapter 8 we find the answer is no!