Contextual Equivalence

[ §5.5, p 44 ]
PCF syntax

Types

\( \tau ::= \text{nat} \mid \text{bool} \mid \tau \rightarrow \tau \)

Expressions

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

\( \mid \text{true} \mid \text{false} \mid \text{zero}(M) \)

\( \mid x \mid \text{if } M \text{ then } M \text{ else } M \)

\( \mid \text{fn } x : \tau \ . \ M \mid M \ M \mid \text{fix}(M) \)

where \( x \in \mathbb{V} \), an infinite set of variables.
PCF syntax

Types

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

Expressions

\[ M ::= 0 \mid \text{succ}(M) \mid \text{pred}(M) \]
\[ \mid \text{true} \mid \text{false} \mid \text{zero}(M) \]
\[ \mid x \mid \text{if } M \text{ then } M \text{ else } M \]
\[ \mid \text{fn } x : \tau . M \mid MM \mid \text{fix}(M) \]

where \( x \in \mathcal{V} \), an infinite set of variables.

Untyped \( \lambda \)-calculus equivalent is \( \text{Y}M \) where

\[ Y = \lambda f. (\lambda x. f(x(x))) (\lambda x. f(x(x))) \]

(Y is not typeable with PCF's simple types)
Two phrases of a programming language are ("Morris style") contextually equivalent ($\approx_{\text{ctx}}$) 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

\[
\text{fix (fn } p : \text{nat} \to \text{nat} \to \text{nat. fn } x : \text{nat. fn } y : \text{nat}
\]
\[
\text{if zero(y) then } x
\]
\[
\text{else succ( p x (pred(y)))) )}
\]
E.g. PCF term for addition

```plaintext
fix (fn p: nat → nat → nat. fn x: nat. fn y: nat
    if zero(y) then pred(succ(x))
    else succ(p x (pred(y))))
```

expect that x and pred(succ x) are contextually equivalent for PCF
PCF contexts $\mathcal{C}$

$$\mathcal{C} ::= - \mid 0 \mid \text{succ}(\mathcal{C}) \mid \text{pred}(\mathcal{C}) \mid \text{zero}(\mathcal{C}) \mid \text{true} \mid \text{false} \mid \text{if } \mathcal{C} \text{ then } \mathcal{C} \text{ else } \mathcal{C} \mid x \mid \text{fn}(x : z \to \mathcal{C} \mid \mathcal{C}) \mid \text{fix}(\mathcal{C})$$

A "hole", or placeholder, to be filled with a PCF term.
PCF contexts $C$

$C ::= - | 0 | \text{succ}(C) | \text{pred}(C) \\
| \text{zero}(C) | \text{true} | \text{false} | \text{if } C \text{ then } C \text{ else } C \\
| x | \text{fn} x : z. C | C C | \text{fix}(C)$

Notation: $C[M] =$ PCF term obtained from $C$ by replacing all occurrences of $-$ by $M$
fix \( (\text{fn } p : \text{nat} \to \text{nat} \to \text{nat}. \text{fn } x : \text{nat}. \text{fn } y : \text{nat} \) \\
if \text{zero}(y) \) then \( \text{pred}(\text{succ}(x)) \) \\
else \( \text{succ}(px(\text{pred}(y))) \) \) \\
is \( C[\text{pred}(\text{succ}(x))] \) for \\
\( C = \text{fix}(\text{fn } p : \text{nat} \to \text{nat} \to \text{nat}. \text{fn } x : \text{nat}. \text{fn } y : \text{nat} \) \\
if \text{zero}(y) \) then \( - \) \\
else \( \text{succ}(px(\text{pred}(y))) \) \)
Two phrases of a programming language are ("Morris style") contextually equivalent ($\equiv_{ctx}$) 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 $\tau$, and a type environment $\Gamma$, the relation $\Gamma \vdash M_1 \equiv_{\text{ctx}} M_2 : \tau$ is defined to hold iff

- Both the typings $\Gamma \vdash M_1 : \tau$ and $\Gamma \vdash M_2 : \tau$ hold.

- For all PCF contexts $C$ for which $C[M_1]$ and $C[M_2]$ are closed terms of type $\gamma$, where $\gamma = \text{nat}$ or $\gamma = \text{bool}$, and for all values $V : \gamma$,

$$C[M_1] \downarrow_\gamma V \iff C[M_2] \downarrow_\gamma V.$$ 

When $\Gamma = \emptyset$, just write $\emptyset \vdash M_1 \equiv_{d\chi} M_2 : \tau$ as $M_1 \equiv_{d\chi} M_2 : \tau$. 

70
Examples of PCF contextual equivalence

\{ x : \text{nat} \} \vdash \text{pred} (\text{succ} (x)) \equiv_{d_x} x : \text{nat}

\{ x : \text{nat} \} \vdash \text{zero} (0) \equiv_{d_x} \text{true} : \text{bool}

? \{ x : \text{nat} \} \vdash \text{zero} (\text{succ} (x)) \equiv_{d_x} \text{false} : \text{bool}
Examples of PCF contextual equivalence

\{ x : \text{nat} \} \vdash \text{pred} (\text{succ} (x)) \equiv_{\Delta x} x : \text{nat}

\{ x : \text{nat} \} \vdash \text{zero} (0) \equiv_{\Delta x} \text{true} : \text{bool}

\{ x : \text{nat} \} \vdash \text{zero} (\text{succ} (x)) \not\equiv_{\Delta x} \text{false} : \text{bool}

because for \( C = (\lambda x : \text{nat}. \neg) \Omega_{\text{nat}} \) we have

\[
\begin{align*}
C[\text{zero} (\text{succ} x)] &= (\lambda x : \text{nat}. \text{zero} (\text{succ} x)) \Omega_{\text{nat}} \\
C[\text{false}] &= (\lambda x : \text{nat}. \text{false}) \Omega_{\text{nat}} \Downarrow_{\text{nat}} \text{false}
\end{align*}
\]
Examples of PCF contextual equivalence

\[ \{x : \text{nat}\} \vdash \text{pred} (\text{succ} (x)) \equiv_{\Delta x} x : \text{nat} \]
\[ \{x : \text{nat}\} \vdash \text{zero} (0) \equiv_{\Delta x} \text{true} : \text{bool} \]
\[ \{x : \text{nat}\} \vdash \text{zero} (\text{succ} (x)) \not\equiv_{\Delta x} \text{false} : \text{bool} \]

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

\[ C [\text{zero} (\text{succ} x)] = (\lambda x : \text{nat}. \text{zero} (\text{succ} x)) \) \( \Omega_{\text{nat}} \not\equiv_{\text{nat}} \]
\[ C [\text{false}] = (\lambda x : \text{nat}. \text{false}) \) \( \Omega_{\text{nat}} \not\equiv_{\text{nat}} \text{false} \]

\underline{MORAL:} easy to show \( \not\equiv_{\Delta x} \) (usually).
Examples of PCF contextual equivalence

\[(\lambda x : \tau \cdot M) M' \equiv_{ctx} M[M'/x] : \tau' \]

(where \(\{\lambda x : \tau \cdot M : \tau \to \tau'\}\)

\(M \equiv_{ctx} \lambda x : \tau \cdot Mx : \tau \to \tau'\)

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

\& \(x \notin \text{fv}(M)\)

\(\text{fix}(M) \equiv_{ctx} M \text{fix}(M) : \tau\)

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

HOW DOES ONE PROVE SUCH FACTS?\)
PCF denotational semantics — aims
PCF denotational semantics — aims

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

\[
\Gamma \vdash M : \tau \quad \mapsto \quad [\Gamma, M : \tau] : [\Gamma] \rightarrow [\tau]
\]

\[
[\Gamma] = [\tau_1] \times \cdots \times [\tau_n]
\]

\text{if } \Gamma = \{ x_1 : \tau_1, \ldots, x_n : \tau_n \}
PCF denotational semantics — aims

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

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

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

• PCF types \( \tau \mapsto \) domains \([\tau]\).

• Closed PCF terms \( M : \tau \mapsto \) elements \([M] \in [\tau]\).
  Denotations of open terms will be continuous functions.

• Compositionality.
  In particular: \([M] = [M'] \Rightarrow [C[M]] = [C[M']]\).

• Soundness.
  For any type \( \tau \), \( M \downarrow_\tau V \Rightarrow [M] = [V] \).
PCF denotational semantics — aims

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

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

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

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

- Adequacy.
  For $\tau = \text{bool}$ or $\text{nat}$, $[M] = [V] \in [\tau] \Rightarrow M \downarrow_\tau V$.  

  (not at function type, because...)
Example 5.6.1

\[ V \triangleq \text{fn } x : \text{nat.} \ (\text{fn } y : \text{nat.} \ y) \ 0 \]

\[ V' \triangleq \text{fn } x : \text{nat.} \ 0 \]

Satisfy:

\[ V \not\rightarrow_{\text{nat} \rightarrow \text{nat}} V' \]

because in general can only prove \( V \Downarrow V' \) for \( V' = V \)
Example 5.6.1

\[
\begin{align*}
V & \triangleq \text{fn } x : \text{nat}. (\text{fn } y : \text{nat}. y)0 \\
V' & \triangleq \text{fn } x : \text{nat}. 0
\end{align*}
\]

Satisfy:

\[
V \notin_{\text{nat} \rightarrow \text{nat}} V'
\]

\[
[ V ] = [ V' ]
\]

because \( (\text{fn } y : \text{nat}. y)0 \downarrow_{\text{nat}} 0 \)

so \( [ (\text{fn } y : \text{nat}. y)0 ] = [0] \) by Soundness
Example 5.6.1

\[
V \triangleq \text{fn } x: \text{nat}. (\text{fn } y: \text{nat}. y) \circ \\
V' \triangleq \text{fn } x: \text{nat}. 0
\]

Satisfy:

\[ V \not\models_{\text{nat} \rightarrow \text{nat}} V' \]

\[ \llbracket V \rrbracket = \llbracket V' \rrbracket \]

because \((\text{fn } y: \text{nat}. y) \circ \downarrow_{\text{nat}} 0\)

so \[ \llbracket (\text{fn } y: \text{nat}. y) \circ \rrbracket = \llbracket 0 \rrbracket \] by Soundness

so \[ \llbracket C[\llbracket (\text{fn } y: \text{nat}. y) \circ \rrbracket] \rrbracket = \llbracket C[0] \rrbracket \] by compositionality

and we can take \( C = \text{fn } x: \text{nat}. - \).
Theorem. For all types $\tau$ and closed terms $M_1, M_2 \in \text{PCF}_\tau$, if $\llbracket M_1 \rrbracket$ and $\llbracket M_2 \rrbracket$ are equal elements of the domain $\llbracket \tau \rrbracket$, then $M_1 \simeq_{\text{ctx}} M_2 : \tau$. 
**Theorem.** For all types $\tau$ and closed terms $M_1, M_2 \in \text{PCF}_\tau$, if $[[M_1]]$ and $[[M_2]]$ are equal elements of the domain $[[\tau]]$, then $M_1 \simeq_{\text{ctx}} M_2 : \tau$.

**Proof.**

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

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

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

and symmetrically (\& similarly for $\Downarrow_{\text{bool}}$). \(\square\)
Proof principle

To prove

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

it suffices to establish

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

To prove

\( M_1 \simeq_{\text{ctx}} M_2 : \tau \)

it suffices to establish

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

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

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

it suffices to establish

$$[M_1] = [M_2]$$ in $$[\tau]$$

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!