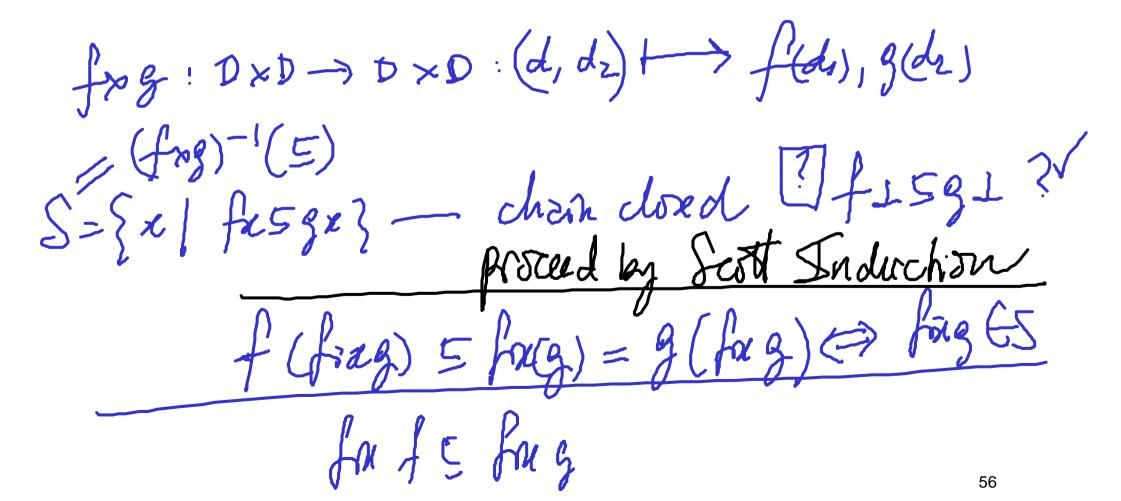
$(f \times d)(d_1 d_2) = (f(d_1), d_2)$ Example (II) (xa): DxD -> DxD Let D be a domain and let  $f, g: D \to D$  be continuous ONG functions such that  $f \circ g \sqsubseteq g \circ f$ . Then,  $f(\bot) \sqsubseteq g(\bot) \rceil \Longrightarrow fix(f) \sqsubseteq fix(g)$ , fresh yfx5 gz (han closed.  $(fxid)^{-1}(\Xi)$ g2ES => f(92) 592 Jdea: S= { 2 [ fr 5 2] ZES => gzES 7625 (fixg) = fizg => fizges fir(f) = fir(g)56

# Example (II)

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

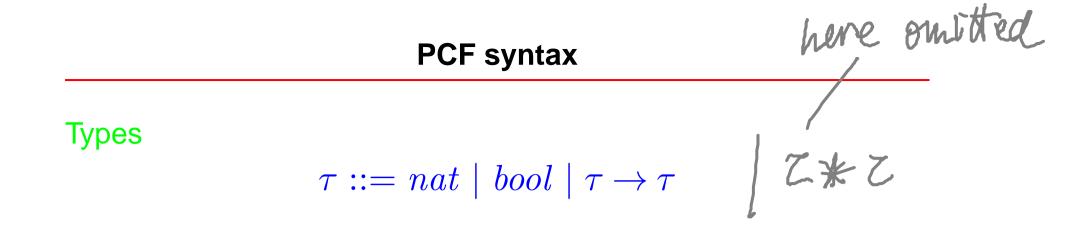
 $f(\perp) \sqsubseteq g(\perp) \implies fix(f) \sqsubseteq fix(g)$ .



NR: txog: DXD-JDXV Woll not work. Example (II) Strengthen fLet D be a domain and let  $f, g: D \to D$  be continuous (fifth fing) functions such that  $f \circ g \sqsubseteq g \circ f$ . Then,  $f(\perp) \sqsubseteq g(\perp) \implies fix(f) \sqsubseteq fix(g)$ . ERCITE. fx =9,y => ffx = 5999 Try Scott Induction 20 missble J= { (xy) | fx 5 gg } fich = h(fah) (fraf) 5 g(fizg) for for fix g 56

# Topic 5

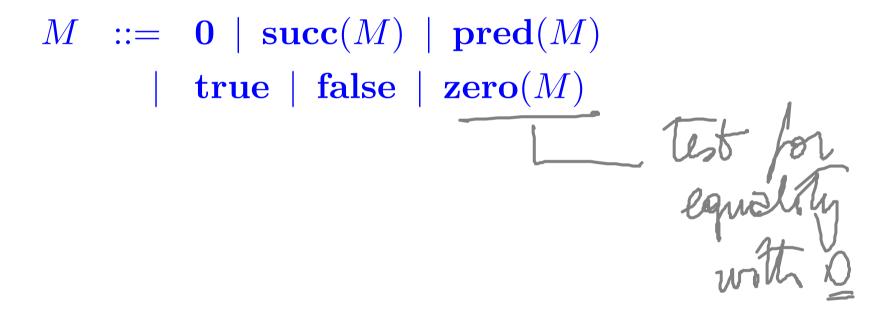
# PCF

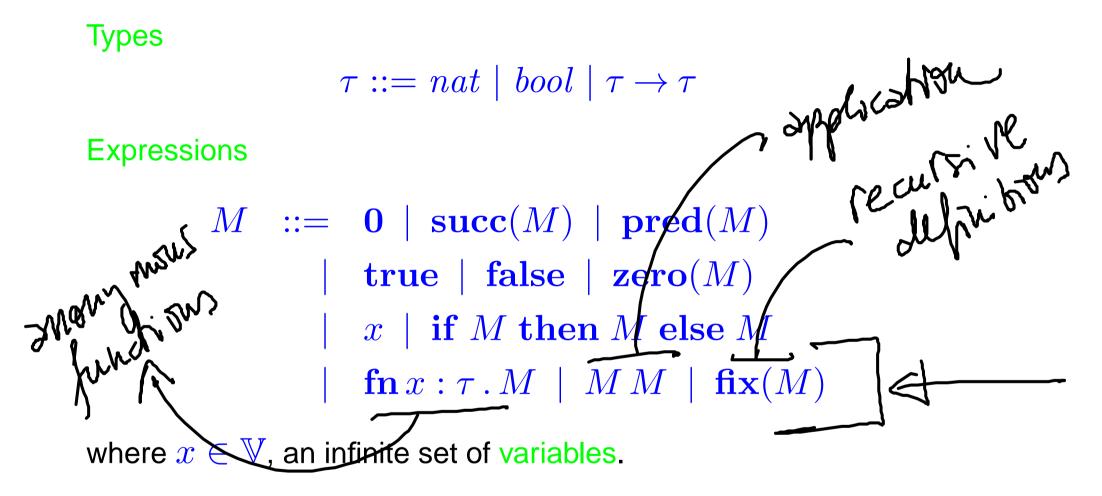


Types

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

Expressions





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

- $\Gamma$  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
- au is a type.

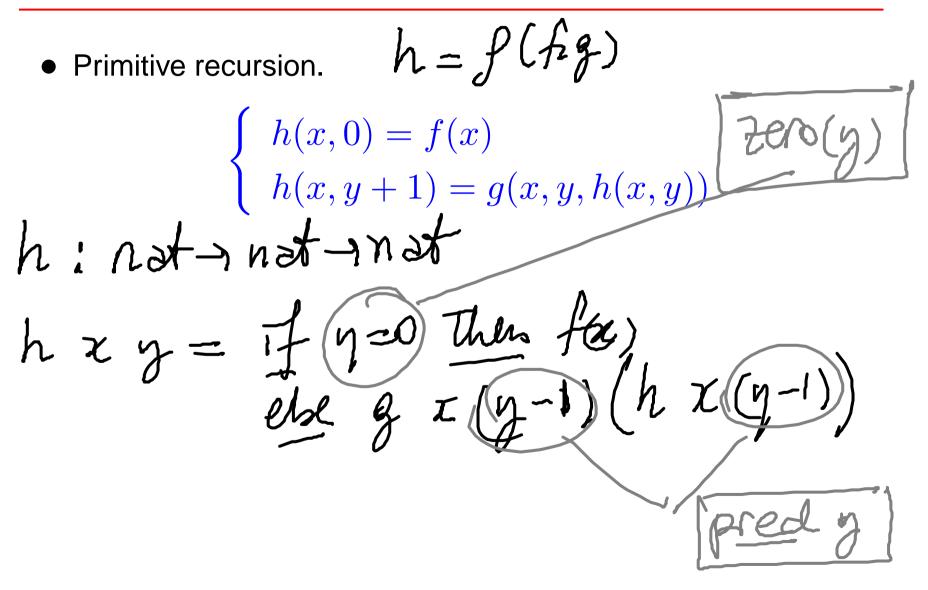
# Notation:

 $M: \tau \text{ means } M \text{ is closed and } \emptyset \vdash M: \tau \text{ holds.}$  $\operatorname{PCF}_{\tau} \stackrel{\operatorname{def}}{=} \{M \mid M: \tau\}.$ 

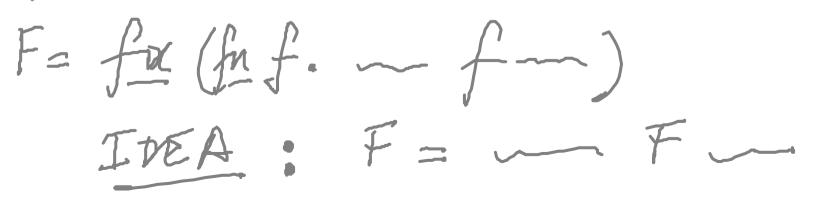
$$(:_{\mathrm{fn}}) \quad \frac{\Gamma[x \mapsto \tau] \vdash M : \tau'}{\Gamma \vdash \mathrm{fn}\, x : \tau \cdot M : \tau \to \tau'} \quad \text{if } x \notin dom(\Gamma)$$

$$\begin{array}{ll} (:_{\mathrm{app}}) & \frac{\Gamma \vdash M_{1}: \tau \rightarrow \tau' & \Gamma \vdash M_{2}: \tau}{\Gamma \vdash M_{1} M_{2}: \tau'} & \text{IDEA}: \\ & & & & \\ & & & \\ (:_{\mathrm{fix}}) & \frac{\Gamma \vdash M: \tau \rightarrow \tau}{\Gamma \vdash \mathrm{fix}(M): \tau} & \text{definition} \\ & & & \\$$

#### **Partial recursive functions in PCF**



 $= \int_{m} h \cdot \int_{m} z \cdot \int_{m} y \cdot \frac{1}{f} (zerog) then f z$ else g z (pred y) (h z (pred y)) f(f,g) = fix(M)



## 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.

misation.  

$$m(x) = \text{ the least } y \ge 0 \text{ such that } k(x, y) = 0$$

$$m(x) = m^{1}(x, 0)$$

$$m^{1}(x, y) = if k(x, y) = 0 \text{ then } y$$

$$if k(x, y) = if k(x, y) = 0 \text{ then } y$$

## **PCF** evaluation relation

takes the form

$$M \Downarrow_{\tau} V$$

where

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

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

$$( \psi_{\text{cbn}} ) \begin{array}{c} \underbrace{M_{1} \psi_{\tau \to \tau'} \text{ fn } x : \tau \cdot M_{1}' \\ \underbrace{M_{1} M_{2} \psi_{\tau'} V} \end{array} \\ ( \psi_{\text{cbn}} ) \begin{array}{c} \underbrace{M_{1} \psi_{\tau \to \tau'} \text{ fn } x : \tau \cdot M_{1}' \\ \underbrace{M_{1} M_{2} \psi_{\tau'} V} \end{array} \\ \end{array}$$

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

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

$$\begin{array}{l} \mathcal{K} \in \mathcal{L} \\ \mathcal{K} \in \mathcal{L} \\ \mathcal{F} (\mathcal{F} ) = \mathcal{F} \\ \mathcal{F} (\mathcal{F} ) = \mathcal{F} \\ \mathcal{F} \\$$

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

## **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 <u>observable results</u> of executing the program. Given PCF terms  $M_1, M_2$ , PCF type  $\tau$ , and a type environment  $\Gamma$ , the relation  $\Gamma \vdash M_1 \cong_{\mathrm{ctx}} M_2 : \tau$  is defined to hold iff

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

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

- PCF types  $\tau \mapsto$  domains  $[\tau]$ .
- 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  $\tau$ ,  $M \Downarrow_{\tau} V \Rightarrow \llbracket M \rrbracket = \llbracket V \rrbracket$ .

• Adequacy.

For  $\tau = bool \text{ or } nat$ ,  $\llbracket M \rrbracket = \llbracket V \rrbracket \in \llbracket \tau \rrbracket \implies M \Downarrow_{\tau} V$ .