# Scott Induction

$\bot \in S$

$S$ chain closed

$$\frac{\forall d \in D \;\; d \in S \Rightarrow f(d) \in S}{\underline{fix}(f) \in S} \quad \frac{S \text{ admiss.}}{f : D \to D}$$

$D$ domain

funct.

# Building chain-closed subsets (III)

**Logical operations:**

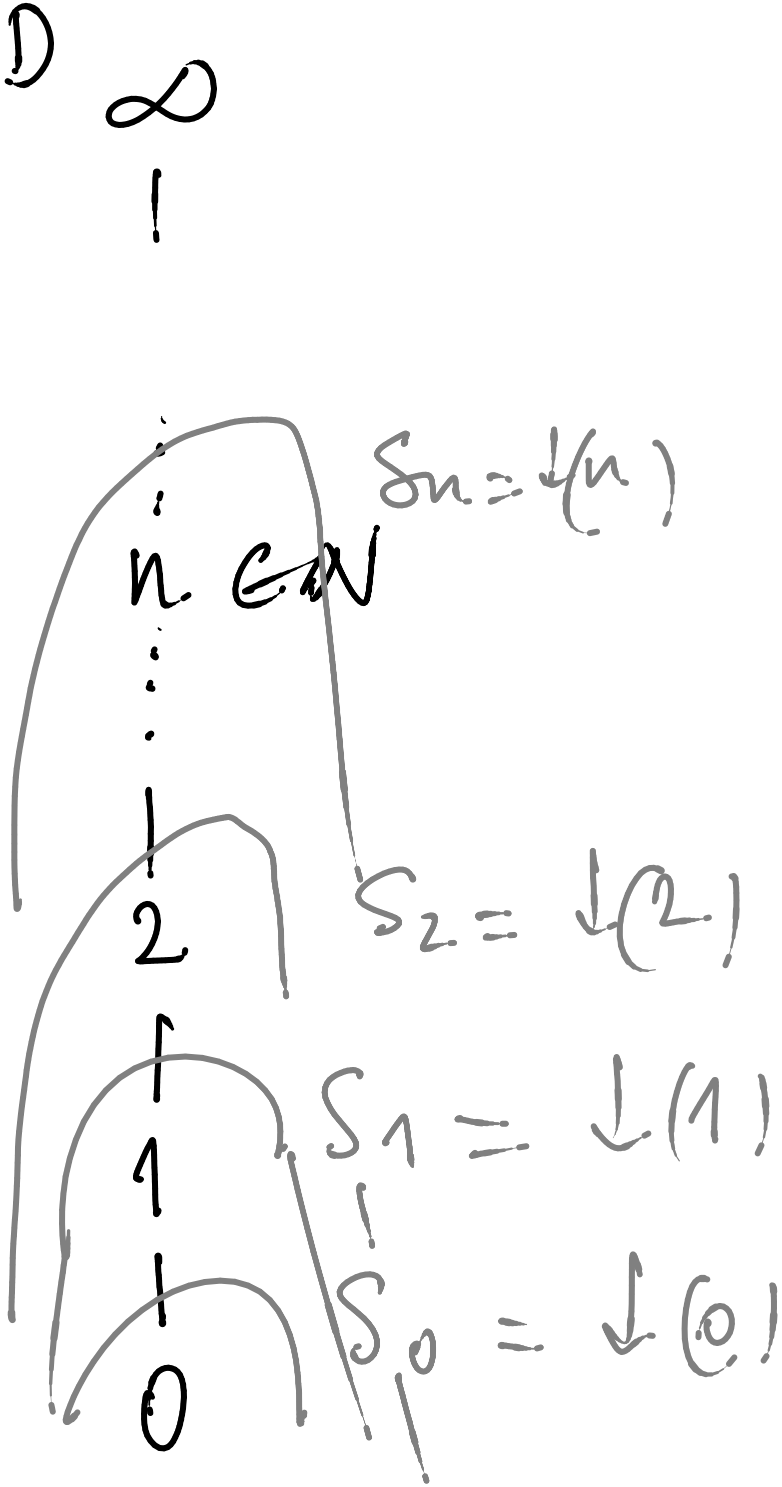
- If $S, T \subseteq D$ are chain-closed subsets of $D$ then

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

  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 $E$.




$$\bigcup_{n\in\mathbb{N}} S_n = D\setminus\{\infty\}$$

## Example (III): Partial correctness

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

$$\mathbf{while}\ X > 0\ \mathbf{do}\ (Y := X * Y; X := X - 1)\ .$$

For all $x, y \geq 0$,

$$\mathcal{F}[X \mapsto x, Y \mapsto y] \downarrow$$
$$\implies \mathcal{F}[X \mapsto x, Y \mapsto y] = [X \mapsto 0, Y \mapsto \cancel{!}x! \cdot y].$$

Recall that

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

where $f : (\mathit{State} \rightharpoonup \mathit{State}) \to (\mathit{State} \rightharpoonup \mathit{State})$ is given by

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

State ⇀ State

*Proof by Scott induction.*

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

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

and show that

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

$\mathrm{fix}(f) \in S$

# *Topic 5*

PCF

# 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.

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

## PCF typing relation, $\Gamma \vdash M : \tau$

- $\Gamma$ 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
- $\tau$ is a type.

**Notation:**

$M : \tau$ means $M$ is closed and $\emptyset \vdash M : \tau$ holds.

$\mathrm{PCF}_\tau \stackrel{\text{def}}{=} \{M \mid M : \tau\}$.

## PCF typing relation (sample rules)

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

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

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

$H\ x\ n = \text{if } (\text{zero } n) \text{ then } F\,x$
$\quad \text{else } G\,x\,(\text{pred } n)\,(H\,x\,(\text{pred } n))$

# Partial recursive functions in PCF

- Primitive recursion.

$$\begin{cases} h(x,0) = f(x) \\ h(x,y+1) = g(x,y,h(x,y)) \end{cases}$$

Given

$F : \tau \to \sigma$

$G : \tau \to nat \to \sigma \to \sigma$

define

$H : \tau \to nat \to \sigma$

$\parallel$

$\underline{\text{fix}}\,(\text{fn } h.\ \text{fn } x.\ \text{fn } n.\ \text{if } (\text{zero } n) \text{ then } F\,x$
$\quad \text{else } G\,x\,(\text{pred } n)\,(h\,x\,(\text{pred } n)))$

F x y = if zero (K x y) then y
else F x (succ y)

M x = F x 0

## Partial recursive functions in PCF

- Primitive recursion.

M = fn x . F x 0

$$\begin{cases} h(x,0) = f(x) \\ h(x,y+1) = g(x,y,h(x,y)) \end{cases}$$

F = ... exercise ...

- Minimisation.

Given

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

K : τ → nat → nat

Define M : τ → nat

## PCF evaluation relation

takes the form

$$M \Downarrow_\tau V$$

where

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

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

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

## PCF evaluation (sample rules)

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

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

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

Is there a value $v:\tau$ s.t. $\underline{\text{fix}}(\text{fn}\ x:\tau.\ x) \Downarrow_\tau v$?

$$\dfrac{\dfrac{\checkmark}{\text{fn}\ x.x \Downarrow \text{fn}\ x.x} \qquad \dfrac{\dfrac{?}{\underline{\text{fix}}(\text{fn}\ x.x)}}{x[\underline{\text{fix}}(\text{fn}\ x.x)/x] \Downarrow \sim}}{\dfrac{(\text{fn}\ x.x)(\underline{\text{fix}}(\text{fn}\ x.x)) \Downarrow \sim}{\underline{\text{fix}}(\text{fn}\ x.x) \Downarrow \sim}}$$

intuitively $\underline{\text{fix}}(\text{fn}\ x:\tau.\ x) = \Omega$ s.t.

$$(\text{fn}\ x:\tau.\ x)(\Omega) \equiv \Omega$$

$$(\text{fn}\ x:\tau.\ x) = \Omega$$

the least such namely $\perp$ denoting non-termination

## 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 $\tau$, and a type environment $\Gamma$, 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 $\gamma$, *where* $\gamma = nat$ *or* $\gamma = bool$, and for all values $V : \gamma$,

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