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.

S, T chain-closed $\Rightarrow$ $S \cup T$ chain-closed.

Consider $d_0 \sqsubseteq d_1 \sqsubseteq \cdots \sqsubseteq d_n \sqsubseteq \cdots$ in $S \cup T$.
$(n \in \mathbb{N})$

(1) $\{d_n\} \cap S$ finite.

Then there is an $N \in \mathbb{N}$ such that

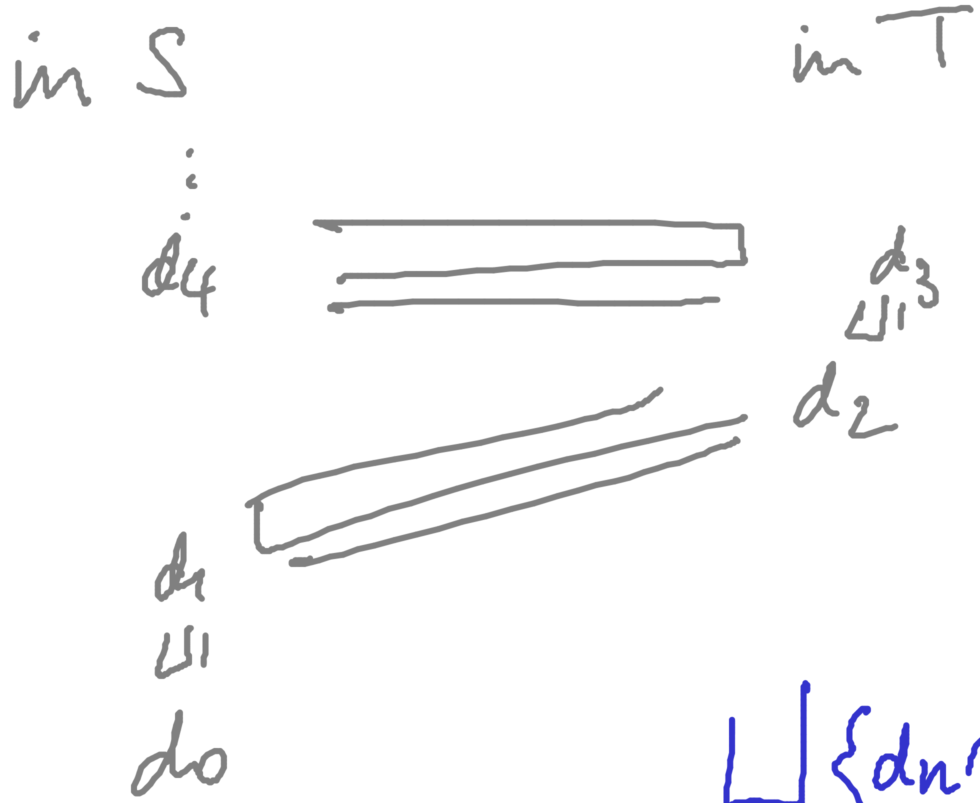
$$d_N \sqsubseteq d_{N+1} \sqsubseteq \cdots \sqsubseteq d_{N+k} \sqsubseteq \cdots \quad \text{in } T$$

and $\bigsqcup_n d_n = \bigsqcup_k d_{N+k} \in T \subseteq S \cup T$

(2) $\{d_n\} \cap T$ finite

Analogous.

(3) $\{d_n\} \cap S$ and $\{d_n\} \cap T$ infinite.



$$\bigsqcup_n \{d_n \cap S\} \in S$$

RTP $\left(\bigsqcup_n d_n\right) \in S \cup T$

$$\bigsqcup_n \{d_n \cap T\} \in T$$

<u>Lemma</u>: $\{d_n\}_n$ $\{e_n\}_n$ in D

such that.

- for every d_n there exists an e_m such that $d_n \sqsubseteq e_m$
- for every e_m there exists a d_n such that $e_m \sqsubseteq d_n$.

Then

$$\bigsqcup_n d_n = \bigsqcup_m e_m$$

.

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 $x, y \geq 0$,

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

Recall that

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

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

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

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

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

$\forall x, y \geq 0$

$$f(w)[X \mapsto x, Y \mapsto y]\downarrow$$

$$\| \quad \begin{cases} (x, y) & x \leq 0 \\ w(x-1, x \cdot y) & x > 0 \end{cases}$$

$\overset{?}{=} (0, x! \cdot y)$

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 α-conversion of bound variables (created by the $\mathbf{fn}$ expression-former): by definition a PCF term is an α-equivalence class of expressions.

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

- Γ 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
- τ is a type.

Notation:

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

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

PCF typing relation (sample rules)

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

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

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

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

Partial recursive functions in PCF

- Primitive recursion.

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

$F : \text{nat} \to \text{nat}$

$G : \text{nat} \to \text{nat} \to \text{nat} \to \text{nat}.$

$H : \text{nat} \to \text{nat} \to \text{nat}.$

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

$F\ x\ n = \text{if}\ (\underline{\text{zero}}\,(K\ x\ n))\ \underline{\text{then}}\ n$
$\underline{\text{else}}\ F\ x\ (\underline{\text{succ}}\ n)$

Partial recursive functions in PCF

- Primitive recursion.

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

$M = \text{fn}.x.\ \underline{\text{fix}}\,(\text{fn}\ f_i\ \text{if}\ (\underline{\text{zero}}\ K\ x\ n)\ \underline{\text{then}}\ n\ \text{else}\ f\ x\,(\underline{\text{succ}}\ n))\,(0)$
fn n

- Minimisation.

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

$M : nat \to nat$

$K : nat \to nat \to nat.$

PCF evaluation relation

takes the form

$$M \Downarrow_\tau V$$

where

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

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

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

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

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

a non-terminating program.

$$\dfrac{\dfrac{fn\ x.x \Downarrow fn\ x.x \qquad x[\underline{fix}(fn\ x.x)/x] \Downarrow}{(fn\ x.x)(\underline{fix}(fn\ x.x)) \Downarrow}}{\underline{fix}(fn\ x.x) = \Omega \Downarrow}$$

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 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 $V : \gamma$,

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