CS 520

Theory of Programming Language

04/28 – 05/05, 2021

An Intro. to Cat. Theory (Continued).

1. Reminder.

① Category. Set, Predom, Dom, Dom., $(P, \sqsubseteq)$.
partial order.

② objects ..... (a) spaces (b) elements in a partial order.

morphisms ~ fns between spaces. generalized version of $\sqsubseteq$ relationship.
structure-pre.

x, y, z.
$x \sqsubseteq y$, $y \sqsubseteq z$,
$\Rightarrow$ $x \sqsubseteq z$.
① ②

functor .... mono. fn.

Cat. fixed-point thm ..... fixed point thm in P.T

(c) objects .... types.
morphisms ... well-typed fns in PL. (functor ... poly. type
nat. trans. ... poly fns.)

③ initial obj., final object, product.

④ Plan: product, co-product, functor, natural transf.

2. Product / Co-Product.

$\mathcal{C}$ ... category, $x, y \in Obj(\mathcal{C})$, $z \in Obj(\mathcal{C})$

① z is a product of x, y if $\boxed{\;\cdots\;}$

② a co-product

$$\begin{tikzcd} & w & \\ & z \arrow[u, dashed, "h"] & \\ x \arrow[ur, "inj_0"] \arrow[uur, bend left, "f"] & & y \arrow[ul, "inj_1"'] \arrow[uul, bend right, "g"'] \end{tikzcd}$$

[$\underset{\text{morphisms}}{\exists}\ \pi_0 : z \to x$, $\pi_1 : z \to y$ s.t. ($inj_0 : x \to z$, $inj_1 : y \to z$)

$\forall\ f : w \to x$, $g : w \to y$ morphisms in $\mathcal{C}$ ($f : x \to w$, $g : y \to w$)

$\underset{\text{unique}}{\exists}\ h : w \to z$ s.t. ($h : z \to w$)

$$\begin{tikzcd} x & & y \\ & z \arrow[ul, "\pi_0"'] \arrow[ur, "\pi_1"] & \\ & w \arrow[u, dashed, "h"] \arrow[uul, bend left, "f"] \arrow[uur, bend right, "g"'] & \end{tikzcd}$$
]

e.g. Predom : P, P' .. predomains, (objects in Predom).

Product of P, P' .... $P \times P' = \{(a,b) \mid a \in P, b \in P'\}$

$(a,b) \sqsubseteq (a',b')$ iff $a \sqsubseteq a'$, $b \sqsubseteq b'$

$\pi_0 : P \times P' \to P$ $\quad$ $\pi_1 : P \times P' \to P'$

$\pi_0(a,b) = a.$ $\quad$ $\pi_1(a,b) = b.$

[Why condition from above holds?

Because $h(\underset{\in w}{c}) \overset{def.}{=} (f(c), g(c)) \in P \times P'$.]

$\langle f, g \rangle$

least upper bound.

$(2^X, \subseteq)$

ex. Co-products in $\widetilde{(P, \sqsubseteq)}$ &

Predom, What are they?

sum of P, P'. $(P+P' = \{\langle 0, a\rangle \mid a \in P\} \cup \{\langle 1, b\rangle \mid b \in P'\}$.

$\langle i, c\rangle \sqsubseteq \langle j, d\rangle$ iff $i=j$ and $c \sqsubseteq d$. $)$

$\underline{[f, g]}$

$inj_0 : P \to P+P'$ $\quad$ $inj_1 : P' \to P+P'$

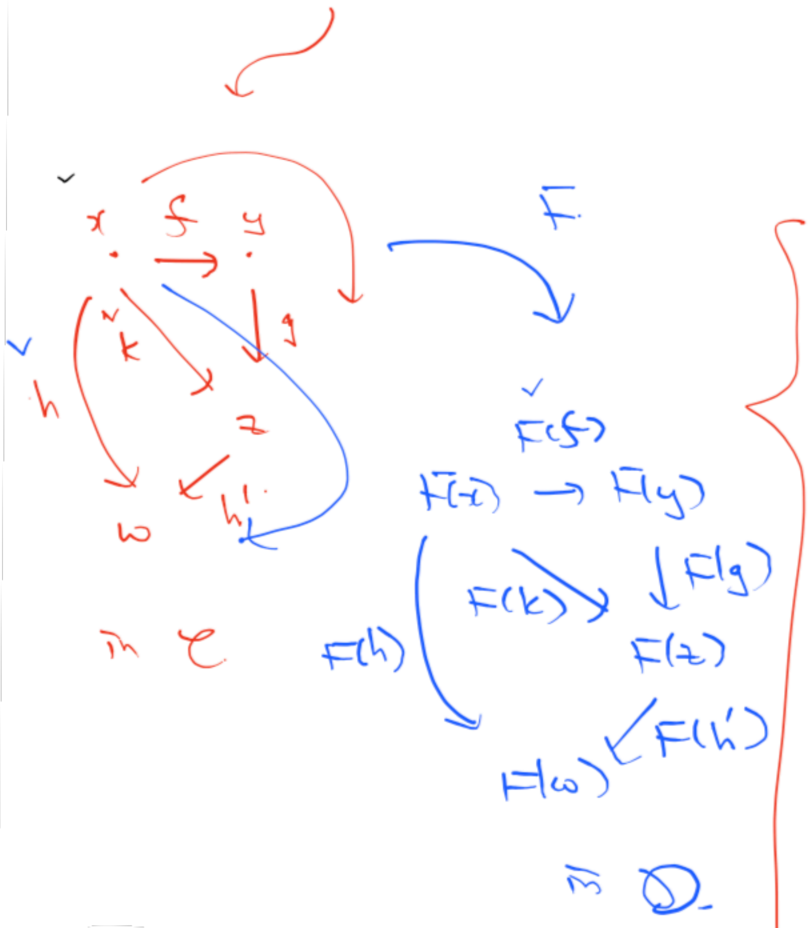
$inj_0(a) = \langle 0, a\rangle$ $\quad$ $inj_1(b) = \langle 1, b\rangle$

$h(\langle 0, a\rangle) = f(a)$ $\quad$ $h(\langle 1, b\rangle) = g(b)$

3. Functor ⋯ map between categories.

$\mathcal{C}, \mathcal{D}$ ⋯ categories.

(1) $F : \mathcal{C} \to \mathcal{D}$ ⋯ functor if $F = \langle F_{obj}, F_{mor} \rangle$ s.t.



(1) $\underset{\sim}{F_{obj}}$ is a map from $Obj(\mathcal{C})$ to $obj(\mathcal{D})$;

(2) $\underline{F_{mor}}$ is a map from $\underset{\text{morphisms of } \mathcal{C}}{\underline{Mor(\mathcal{C})}}$ to $Mor(\mathcal{D})$;

(3) they satisfy the following three conditions:

- $\forall$ morphism $f : x \to y$ in $\mathcal{C}$, ↱ type check.

  $F_{mor}(f)$ is a morphism from $F_{obj}(x)$ to $F_{obj}(y)$ in $\mathcal{D}$. (in notation,

  $\underset{\text{morphism in } \mathcal{D}}{F(f) : F(x) \to F(y)}$)

- F should preserve identity morphism

  $$\underset{Hom_{\mathcal{C}}[x,x]}{F(id_x)} = \underset{Hom_{\mathcal{D}}[F(x), F(x)]}{id_{F(x)}}$$

- F should preserve morph. composition.

  $\forall f : x \to y,\ g : y \to z$ morphisms in $\mathcal{C}$.

  $$F(g \underset{\mathcal{C}}{\circ} f) = F(g) \underset{\mathcal{D}}{\circ} F(f)$$

② e.g. $\mathcal{C} = (2^X, \subseteq)$

$\mathcal{D} = (2^Y, \subseteq)$. F is a functor iff F_{obj} is a monotone fn.

(1) Constant functor.

$F_{\mathbb{Z}} : \mathbf{Predom} \to \mathbf{Predom}$

$F_{\mathbb{Z}}(P) = \mathbb{Z}$

$F_{\mathbb{Z}}(f) = id_{\mathbb{Z}}$.

(2).

$Id : \mathbf{Predom} \to \mathbf{Predom}$.

$Id(P) = P$

$Id(f) = f$.

(3) $-_{\perp} : \mathbf{Predom} \to \mathbf{Predom}$.

$(-_{\perp})(P) = P_{\perp}$

$$(-_{\perp})(f)(a) = \begin{cases} f(a) & \text{if } a \neq \perp \\ \perp & \text{otherwise} \end{cases}$$

(3) Product / Co-Product.

$$\left.\begin{array}{l} F_1 : \mathbf{Predom} \to \mathbf{Predom} \\ F_2 : \mathbf{Predom} \to \mathbf{Predom} \end{array}\right\} \text{given.}$$

$F : \mathbf{Predom} \to \mathbf{Predom}$.

$$F(P) = F_1(P) \times F_2(P)$$

$$F(f) = \underbrace{F_1(f)}_{F_1(P) \to F_1(P')} \times \underbrace{F_2(f)}_{F_2(P) \to F_2(P')}$$

$f : P \to P'$

$$\overline{}$$

$$F_1(P) \times F_2(P) \to F_1(P') \times F_2(P')$$

$$(a, b) \mapsto (F_1(f)(a), F_2(f)(b)).$$

$G : \mathbf{Predom} \to \mathbf{Predom}$.

$$G(P) = F_1(P) + F_2(P)$$

$$G(f) = \underbrace{F_1(f) + F_2(f)}.$$

$$F_1(P) + F_2(P) \to F_1(P') + F_2(P')$$

$$\langle i, c\rangle \mapsto \begin{cases} \langle i, F_1(f)(c)\rangle & \text{if } i = 0. \\ \langle i, F_2(f)(c)\rangle & \text{if } i = 1. \end{cases}$$

(4). $\Omega \cong [\hat{\Sigma} + \mathbb{Z} \times \Omega + \cancel{\cdots}]_\perp$

constant functors.

$$F(\Omega) = [\hat{\Sigma} + \mathbb{Z} \times \Omega]_\perp$$

$$F(f) : F(\Omega) \to F(\Omega')$$

$f : \Omega \to \Omega'$

functors are used in our recursive domain equations.

4. Natural Transformation. .... map between functors.

[Domain theory : $f, g : P \xrightarrow[m]{} P'$.

$f \sqsubseteq g$ ~~iff~~ $f(x) \sqsubseteq g(x)$ for all x.

PL : $F, G : \mathcal{C} \to \mathcal{D}$. ... type operators.

$\eta : F \xrightarrow{\cdot} G$. ... polymorphic fns.]

$F, G : \mathcal{C} \to \mathcal{D}$ ... functors.

A natural transformation $\eta : F \xrightarrow{\cdot} G$ is a family of morphisms in $\mathcal{D}$ indexed by objects in $\mathcal{C}$, denoted $\{\eta_x\}_{x \in \mathrm{Obj}(\mathcal{C})}$, s.t.

type

(1) $\eta_x : F(x) \to G(x)$ in $\mathcal{D}$, (... typecheck)

(2) $\forall f : x \to y$ in $\mathcal{C}$ (morphism in $\mathcal{C}$). in $\mathcal{D}$: the following diagram commutes in $\mathcal{D}$:

$$\begin{array}{ccc} F(x) & \xrightarrow{\eta_x} & G(x) \\ {\scriptstyle F(f)}\downarrow & & \downarrow{\scriptstyle G(f)} \\ F(y) & \xrightarrow{\eta_y} & G(y) \end{array}$$

in $\mathcal{D}$.

naturality condition.

co-product, product ... Freedom.

$\mathrm{inj}_0, \mathrm{inj}_1$. π_0, π_1 ..... natural transformations.