CS520 Lecture 7
An Introduction to Category Theory

March 22, 2020

7.1 Motivation

1. The category theory is a branch of mathematics that studies essentially same or common notions and principles in different areas of mathematics. It is very abstract, but provides good guidelines about finding right definitions and right or meaningful questions to ask in a new mathematical theory.
2. The category theory had huge influence on the research on programming languages and the development of practical programming languages, such as Scala, Haskell, and Rust.
3. In this course, we will focus on the influence of the category theory on the semantics research. We will study abstract categorical concepts that give rise to popular notions appearing in the semantics of programming languages. Another big objective is to understand a result in the category theory (or a categorical formulation of domain theory) that ensures the existence of certain recursively defined domains, such as the following Ω that you saw in the chapter 5 of Reynolds' book

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

$$\hat{\Sigma} \stackrel{def}{=} \Sigma + \Sigma$$

4. We will study only a tiny part of the category theory. If you are excited about it and are willing to read a math book, I recommend Mac Lane's "Categories for the working mathematicians".

7.2 Definition of Category

Definition 1
A category C is a tuple $(Obj, Hom, \circ, id)$ where

1. *Obj is a collection of elements called objects.*
2. *for all objects $x, y \in Obj$, $Hom[x, y]$ is a collection of elements called morphisms from x to y.*
3. *for all objects $x, y, z \in Obj$, $\circ_{x,y,z}$ (or simply $\circ$) is a map from $Hom[y, z] \times Hom[x, y]$ to $Hom[x, z]$ and is called composition*
4. *for every object $x \in Obj$, id_x is an element in $Hom[x, x]$ and is called identity morphism*
5. *these data should satisfy associativity and identity axioms*

$$\begin{array}{ll} [associativity] & [identity] \\ \forall w, x, y, z \in Obj & \forall x, y \in Obj \\ \forall f \in Hom[w, x] & \forall f \in Hom[x, y] \\ \forall g \in Hom[x, y] & f \circ id_x = id_y \circ f = f \\ \forall h \in Hom[y, z] & \\ h \circ (g \circ f) = (h \circ g) \circ f & \end{array}$$

1. Although not perfect, a reasonably good intuition is that a category C is a collection of spaces that you encounter in mathematics, such as metric spaces, vector spaces, topological spaces, etc. The Obj part of C consists of spaces, and the $Hom[x, y]$ part of C consists of maps between spaces that preserve the structure of the spaces. $\circ$ is then the usual function composition and id_x is the identity function on x.
2. Here are some examples that match the intuition that I just explained.
(1) Set $\cdots$ Category of sets and functions

- obj is the collection of all sets.
- $Hom[x, y]$ is the set of all functions from x to y.
  (Note that since $x, y \in Obj$, they are sets)
- $\circ$ is the function composition.
- id_x is the identity function on x.

(2) Predom $\cdots$ Category of predomains and continuous functions.

- obj is the collection of all predomains.
- $Hom[x, y]$ is the set of all continous functions from x to y.
- $\circ$ is the function composition.
- id_x is the identity function on x.

(3) Dom $\cdots$ Category of domains and continuous functions.

- obj is the collection of all domains.
- $Hom[x, y]$ is the set of all continuous functions from x to y.
- $\circ$ is the function composition.
- id_x is the identity function on x.

Note that Dom is in a sense included in Predom. (Technically, it is a full subcategory of Predom).
3. As indicated by my use of the phrase "not perfect", there are categories that do not match the intuition well. Here is a very well-known example.
(1) Let $(X, \sqsubseteq)$ be a partially ordered set. It can be understood as a category:

- obj = X
- $Hom[x, y] = \begin{cases} \emptyset & \text{if } x \not\sqsubseteq y \\ \{*\} \text{ if } x \sqsubseteq y \end{cases}$ (a set with one element. It doesn't matter what that element is)
- $id_x = *$

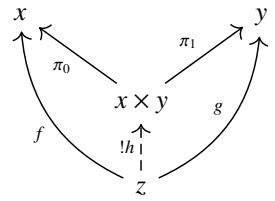
- $\circ_{x,y,z} \in [Hom[y,z] \times Hom[x,y] \to Hom[x,z]]$
  If $x \sqsubseteq y$ and $y \sqsubseteq z$ (i.e. $Hom[x,y] \neq \emptyset$ and $Hom[y,z] \neq \emptyset$), then $\circ_{x,y,z}$ is the constant function to the unique element in $Hom[x,z]$. ($Hom[x,z] \neq \emptyset$ in this case because of the transitivity of $\sqsubseteq$)
  Otherwise, (i.e., $Hom[x,y] = \emptyset$ or $Hom[y,z] = \emptyset$), $\circ_{x,y,z}$ is the empty function (the function whose graph is the empty set).

4. In the category theory, we often use commutative diagrams to express the equality of two morhpisms. For instance, $\begin{array}{ccc} x & \xrightarrow{f} & y \\ \downarrow{\scriptstyle k} & & \downarrow{\scriptstyle g} \\ a & \xrightarrow{h} & z \end{array}$ expresses that x, y, a, z are objects in a category, and f, g, h, k are morphisms with domains and codomains indicated by the arrows (for instance, $f \in Hom[x,y]$), and $\underbrace{g \circ f = h \circ k}_{\text{the most important bit}}$

5. Another intuition about categories is that objects in a category are types in a programming languages and morphisms from x to y are functions in the language from the input of type x to the output of type y.

7.3 Initial and Terminal Objects, Product and Co-Product (or Sum)

1. In a category C, we can build a new object out of existing ones. Often this new object sattisfies one of the well-known conditionsm and has a well-known name. We will study a few such names.
2. Consider a category C and its objects x, y.
(1) An object z is a product of x and y, often written as $z = x \times y$, if there are morphisms $\underbrace{\pi_0 \in Hom[z,x]}_{\text{often written as } \pi_0 : x \times y \to x}$ and $\underbrace{\pi_1 \in Hom[z,y]}_{\text{often written as } \pi_1 : x \times y \to y}$ s.t. for all objects w and morphisms $f : w \to x$ and $g : w \to y$, there exists a unique morhpism h[1] $: w \to x \times y$ with $f = \pi_0 \circ h$ and $g = \pi_1 \circ h$.

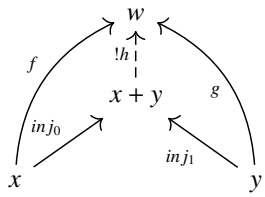


! means uniqueness.

$\cdots$ means existence.

(2) An object z is a co-product (or sum) of x and y, often written as $z = x + y$, if there are morphisms $inj_0 : x \to x + y$ and $inj_1 : y \to x + y$ s.t. for all objects w and morhpisms $f : x \to w$ and $g : y \to w$, there exists a unique morphism $h^2 : x + y \to w$ with $f = h \circ inj_0$ and $g = h \circ inj_1$.

[1] we often write h as $\langle f, g \rangle$. Reynolds writes it as $f \otimes g$
[2] often written as $[f, g]$. Reynolds writes it as $f \oplus g$



(3) An object x is initial if for every object w, there exists the unique morphism $h : x \to w$.
(4) An object x is final if for every object w, there exists the unique morhpism h[3] $: w \to x$.
3. Note that all of these notions are defined purely in terms of morphisms, without referring to elements of objects. This is a bit like specifying a property of an abstract data type in terms of its operations, not in terms of its implementatio(n.
4. In the category Predom, the product of two predomains P_0 and P_1 that we studied in Chap 5 is indeed a categorical product in (1). Also, the sum of P_0 and P_1 in Chap 5 is indeed a categorical sum or co-product. The predomain of the singleton set $(\{*\}, \sqsubseteq)$ is a terminal object. The predomain of the empty set $(\{\}, \sqsubseteq)$ is an initial object.
5. Consider the category corresponding to a partially ordered set $(X, \sqsubseteq)$. Then, an object $x \in X$ is initial if and only if it is the least element in X. It is final if and only if it is the greatest element. For objects x, y in X, $x + y$ is the least upper bound of x and y, and $x \times y$ is the greatest lower bound of x and y.
Exercise Prove 4. and 5.

7.4 Functor and Natural Transformation

1. Intuitively, a functor is a structure-preserving map from one category to another. A natural transformation is a uniform map from one functor to another. In PL terms, a functor is a type constructor. (Recall that in this analogy, an object in a category is a type). And a natural transformation is a polymorphic function.
2. Let $\mathcal{C}$ and $\mathcal{D}$ be categories.

Definition 2
A functor F from $\mathcal{C}$ to $\mathcal{D}$ is a pair of two maps F_{obj} and F_{mor} s.t.

1. *F_{obj} map $\underbrace{\text{objects in } \mathcal{C}}_{\text{objects of } \mathcal{C}}$ to objects in $\mathcal{D}$.*
2. *for objects $x, y \in Obj(\mathcal{C})$*

$$(F_{mor})_{x,y} \in [\underbrace{Hom_{\mathcal{C}}[x, y]}_{\text{morphisms in } \mathcal{C}} \to \underbrace{Hom_{\mathcal{D}}[F_{obj}(x), F_{obj}(y)]}_{\text{morphisms in } \mathcal{D}}]$$

3. *F_{mor} preserves $\circ$ and id:*

$$\text{for all objects } x \text{ of } \mathcal{C}, (F_{mor})_{x,x}(id_x) = id_{F_{obj}(x)}$$

[3] often written as $!w$ or !

$$\textit{for all morhpisms } f \in Hom_C[x,y] \textit{ and } g \in Hom_C[y,z],$$
$$(F_{mor})_{x,z}(g \circ f) = (F_{mor})_{y,z}(g) \circ (F_{mor})_{x,y}(f)$$

we use F to mean F_{obj} and $(F_{mor})_{x,y}$

3. To see common examples, we need to understant the product of two categories. When $\mathcal{C}$ and $\mathcal{D}$ are categories, the product category $\mathcal{C} \times \mathcal{D}$ is defined as follows

- $obj = \{(x,y) | x \in Obj(\mathcal{C}) \wedge y \in Obj(\mathcal{D})\}$
- $Hom_{\mathcal{C}\times\mathcal{D}}[(u,v),(x,y)] = \{(f,g) | f \in Hom_{\mathcal{C}}[u,x] \wedge g \in Hom_{\mathcal{D}}[v,y]\}$
- $(f,g) \circ (f',g') = (f \circ f', g \circ g')$
- $id_{(x,y)} = (id_x, id_y)$

4. The $\times, +, \perp$ operators on predomains are in fact functors.

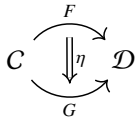
$$(-)_\perp : Predom \to Predom$$
$$(-)_\perp(P) = P_\perp$$
$$(-)_\perp(f) = \lambda x. \begin{cases} \perp & \text{if } x = \perp \\ f(x) & \text{if } x \neq \perp \end{cases}$$
$$\times^4 : \underbrace{Predom \times Predom}_{\text{Product of categories}} \to Predom$$
$$\times(P_0, P_1) = \underbrace{P_0 \times P_1}_{\text{Product of predomains}}$$
$$\times(f,g) = \lambda(x,y).(f(x), g(y)) = f \times g^5$$
$$+^6 : Predom \times Predom \to Predom$$
$$+(P_0, P_1) = \underbrace{P_0 + P_1}_{\text{sum of predomains}}$$
$$+(f,g) = \lambda x. \begin{cases} inj_0(f(u)) & \text{if } x = inj_0(u) \\ inj_1(g(v)) & \text{if } x = inj_1(v) \end{cases} = f + g^7$$

5. One other Important functor is the identity functor:

$$Id : Predom \to Predom$$
$$Id(X) = X$$
$$Id(f) = f$$

6. Let F, G be functors from $\mathcal{C}$ to $\mathcal{D}$.

Definition 3

A natural transformation η from F to G denoted $\eta : F \dot{\to} G : \mathcal{C} \to \mathcal{D}$ or 

is a collection of morphisms in $\mathcal{D}$ indexed by objects in $\mathcal{C}$

$$(\eta_x : F(x) \to G(x)))x \in Obj(\mathcal{C})$$

such that

1. *for each object x in $\mathcal{C}$, η_x is a $\mathcal{D}$-morphism form $F(x)$ to $G(x)$*
2. *for every morphism $f : x \to y$ in $\mathcal{C}$ (i.e. $f \in Hom_{\mathcal{C}}[x, y]$),*

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

This is called naturality condition

7. One intuition is that F and G are type constructors, and η is a polymorphic function from F to G. Whenver we are given a type x, we have an instantiation of η at x that is a function from the type $F(x)$ to the type $G(x)$. The condition 1 says that η should typecheck. The condition 2 says that what η does at x should be identical in a sense to what it does at y. In other words, η should not depend much on its type parameter x. This is called uniformity.
8. Here are a few examples.

$$\begin{aligned} &unit : id \dot{\to} (-)_\perp : Predom \to Predom \\ &unit_P \in [P \to P_\perp] \\ &unit_P(x) = x \end{aligned}$$

Let Fst be a functor from $\mathcal{C} \times \mathcal{D}$ to $\mathcal{C}$. defined by

$$\begin{aligned} Fst(x, y^8) &= x \\ Fst(f, g^9) &= f \end{aligned}$$

Then,

$$\begin{aligned} &\pi_0 : (-) \times (-) \dot{\to} Fst : Predom \times Predom \to Predom \\ &(\pi_0)_{P,P'} \in [P \times P' \to_C P] \\ &(\pi_0)_{P,P'}(a, b) = a \end{aligned}$$

<u>exercise</u> Show that unit and π_0 are indeed natural transformation.
<u>exercise</u> $\pi_1, in j_0, in j_1$ are also natrual transformations if we pick appropriate functors. Find such functors.
9. Notice that all of $unit, \pi_0, \pi_1, in j_0, in j_1$ do something very straightforward intuitively. They don't do any clever tricks. Naturality condition says in a sense that a natural transformation doesn't do anything clever. In more positive terms, it says that a natural transformation only performs canonical operations.