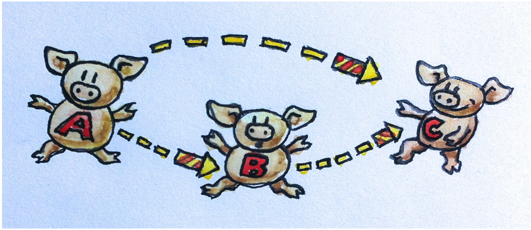
Category Theory
& Functional Data Abstraction

Brandon Shapiro

Math 100b



Categories

- A category **C** is a collection of objects with arrows (often called morphisms) pointing between them
- $Hom_{\mathbf{C}}(X, Y)$ is the set of morphisms in **C** from X to Y
- If $f \in Hom_{\mathbf{C}}(X, Y)$ and $g \in Hom_{\mathbf{C}}(Y, Z)$, then there exists a morphism $f \circ g$ in $Hom_{\mathbf{C}}(X, Z)$ (composition is associative)
- For every object X in **C**, there is an identity morphism $1_X \in Hom_{\mathbf{C}}(X, X)$ ($f \circ 1_X = f$ and $1_X \circ g = g$)

$$
\begin{tikzcd}
A \arrow[r, "f"] \arrow[dr, "g\circ f"'] \arrow[loop left, "1_A"] & B \arrow[d, "g"] \arrow[loop right, "1_B"] \\
& C \arrow[loop right, "1_C"]
\end{tikzcd}
$$

Categories

- A category **C** is a collection of objects with arrows (often called morphisms) pointing between them
- $Hom_{\mathbf{C}}(X, Y)$ is the set of morphisms in **C** from X to Y
- If $f \in Hom_{\mathbf{C}}(X, Y)$ and $g \in Hom_{\mathbf{C}}(Y, Z)$, then there exists a morphism $f \circ g$ in $Hom_{\mathbf{C}}(X, Z)$ (composition is associative)
- For every object X in **C**, there is an identity morphism $1_X \in Hom_{\mathbf{C}}(X, X)$ ($f \circ 1_X = f$ and $1_X \circ g = g$)

$$
\begin{tikzcd}
A \arrow[r, "f"] \arrow[dr, "g \circ f"'] \arrow[loop left, "1_A"] & B \arrow[d, "g"] \arrow[loop right, "1_B"] \\
& C \arrow[loop right, "1_C"]
\end{tikzcd}
$$

Examples

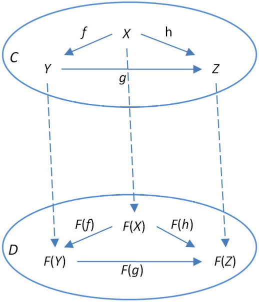
- **Set** is the category of all sets, with functions between sets as the morphisms
- All groups also form a category, **Grp**, with group homomorphisms as its morphisms
- **Ring** and R-**mod** for some ring R can be formed with ring and module homomorphisms as morphisms
- A subcategory of category **C** is a category with all of its objects and morphisms contained in **C**
- Finite sets and the functions between them form a subcategory of **Set**, and abelian groups are a subcategory of **Grp**. Fields form a subcategory of the category of commutative rings, which is itself a subcategory of **Ring**

Examples

- **Set** is the category of all sets, with functions between sets as the morphisms
- All groups also form a category, **Grp**, with group homomorphisms as its morphisms
- **Ring** and *R*-**mod** for some ring *R* can be formed with ring and module homomorphisms as morphisms
- A subcategory of category **C** is a category with all of its objects and morphisms contained in **C**
- Finite sets and the functions between them form a subcategory of **Set**, and abelian groups are a subcategory of **Grp**. Fields form a subcategory of the category of commutative rings, which is itself a subcategory of **Ring**

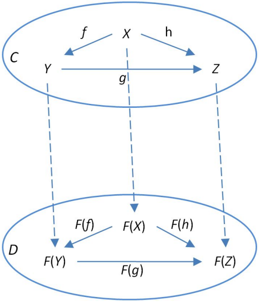
Functors

- A functor is a structure preserving map between categories
- For categories **C** and **D**, a covariant functor $\mathcal{F} : \mathbf{C} \to \mathbf{D}$ sends the objects of **C** to objects in **D**, and sends the morphisms in **C** to morphisms in **D**
- If $f \in Hom_{\mathbf{C}}(X, Y)$, $\mathcal{F}(f) \in Hom_{\mathbf{D}}(\mathcal{F}(X), \mathcal{F}(Y))$
- $\mathcal{F}(1_X) = 1_{\mathcal{F}(X)}$, $\mathcal{F}(f \circ g) = \mathcal{F}(f) \circ \mathcal{F}(g)$



Functors

- A functor is a structure preserving map between categories
- For categories **C** and **D**, a covariant functor $\mathcal{F} : \mathbf{C} \to \mathbf{D}$ sends the objects of **C** to objects in **D**, and sends the morphisms in **C** to morphisms in **D**
- If $f \in Hom_{\mathbf{C}}(X, Y)$, $\mathcal{F}(f) \in Hom_{\mathbf{D}}(\mathcal{F}(X), \mathcal{F}(Y))$
- $\mathcal{F}(1_X) = 1_{\mathcal{F}(X)}$, $\mathcal{F}(f \circ g) = \mathcal{F}(f) \circ \mathcal{F}(g)$



Examples

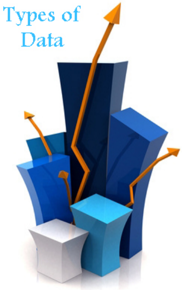
- The identity functor from **C** to **C** sends every object and morphism in **C** to itself.
- Let $\mathcal{F}$ be a map from **Grp** to **Set** sending groups and homomorphisms in **Grp** to themselves in **Set**. $\mathcal{F}$ is a functor from **Grp** to **Set** called the 'forgetful functor'
- Similarly, forgetful functors exist from **Ring** and R-**mod** to **Grp** and to **Set**
- A functor from a category to itself is called an endofunctor
- The identity functor is an endofunctor

Examples

- The identity functor from **C** to **C** sends every object and morphism in **C** to itself.
- Let $\mathcal{F}$ be a map from **Grp** to **Set** sending groups and homomorphisms in **Grp** to themselves in **Set**. $\mathcal{F}$ is a functor from **Grp** to **Set** called the ‘forgetful functor’
- Similarly, forgetful functors exist from **Ring** and R-**mod** to **Grp** and to **Set**
- A functor from a category to itself is called an endofunctor
- The identity functor is an endofunctor

Date Types

- In computer programming languages, a data type is a set of elements that can be represented by a computer (finitely in binary) in the same way
- Two of the most common data types are $\mathbb{Z}$ and $\mathbb{R}$
- Real-world computing has constraints on memory, etc.
- Mathematically, a data type can be treated just as a set



Maybe

- **Set** has sets as objects and functions as morphisms

$$\mathcal{M}aybe : \mathbf{Set} \to \mathbf{Set}$$
$$\mathcal{M}aybe(A) = A \cup \{Nothing\}$$

- $\mathcal{M}aybe$ lets us define 'safe' versions of partial functions

$$f : \mathbb{R} \to \mathcal{M}aybe(\mathbb{R})$$
$$f(0) = Nothing$$
$$f(x) = 1/x \ (x \neq 0)$$

Maybe

- **Set** has sets as objects and functions as morphisms

$$\mathcal{M}aybe : \mathbf{Set} \to \mathbf{Set}$$
$$\mathcal{M}aybe(A) = A \cup \{Nothing\}$$

- $\mathcal{M}aybe$ lets us define 'safe' versions of partial functions

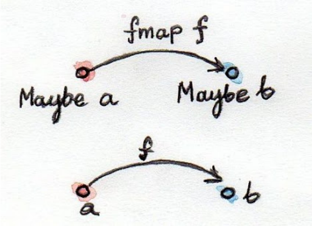
$$f : \mathbb{R} \to \mathcal{M}aybe(\mathbb{R})$$
$$f(0) = Nothing$$
$$f(x) = 1/x \ (x \neq 0)$$

Maybe

- $\mathcal{M}aybe$ is a functor from **Set** to **Set** (endofunctor)
- Needs a mapping for the morphisms (functions)

$$\mathcal{M}map : Hom(A, B) \to Hom(\mathcal{M}aybe(A), \mathcal{M}aybe(B))$$
$$\mathcal{M}map(f)(Nothing) = Nothing$$
$$\mathcal{M}map(f)(x) = f(x) \; (x \neq Nothing)$$

- $\mathcal{M}map(1_A) = 1_{\mathcal{M}aybe(A)}$
- $\mathcal{M}map(f \circ g) = \mathcal{M}map(f) \circ \mathcal{M}map(g)$

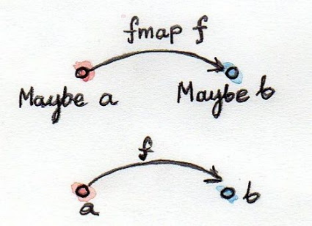


Maybe

- $\mathcal{M}aybe$ is a functor from **Set** to **Set** (endofunctor)
- Needs a mapping for the morphisms (functions)

$$\mathcal{M}map : Hom(A, B) \to Hom(\mathcal{M}aybe(A), \mathcal{M}aybe(B))$$
$$\mathcal{M}map(f)(Nothing) = Nothing$$
$$\mathcal{M}map(f)(x) = f(x) \ (x \neq Nothing)$$

- $\mathcal{M}map(1_A) = 1_{\mathcal{M}aybe(A)}$
- $\mathcal{M}map(f \circ g) = \mathcal{M}map(f) \circ \mathcal{M}map(g)$



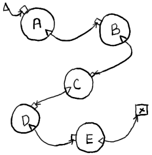
List

- $\mathcal{L}ist$ sends a set A to the set of 'lists' of elements in A

$$\mathcal{L}ist : \mathbf{Set} \to \mathbf{Set}$$
$$\mathcal{L}ist(A) = \{()\} \cup \{(x, xlist) | x \in A, xlist \in \mathcal{L}ist(A)\}$$

- () is called the empty list

$$(1, (2, (3, (4, ())))) \in \mathcal{L}ist(\mathbb{Z})$$
$$(1/2, (Nothing, (1/4, ()))) \in \mathcal{L}ist(\mathcal{M}aybe(\mathbb{Q}))$$
$$(1, 2, 3, 4) \in \mathcal{L}ist(\mathbb{Z})$$



Linked List

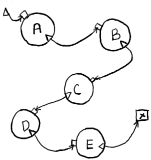
List

- $\mathcal{L}ist$ sends a set A to the set of 'lists' of elements in A

$$\mathcal{L}ist : \mathbf{Set} \to \mathbf{Set}$$
$$\mathcal{L}ist(A) = \{()\} \cup \{(x, xlist) | x \in A, xlist \in \mathcal{L}ist(A)\}$$

- $()$ is called the empty list

$$(1, (2, (3, (4, ())))) \in \mathcal{L}ist(\mathbb{Z})$$
$$(1/2, (Nothing, (1/4, ()))) \in \mathcal{L}ist(\mathcal{M}aybe(\mathbb{Q}))$$
$$(1, 2, 3, 4) \in \mathcal{L}ist(\mathbb{Z})$$



Linked List

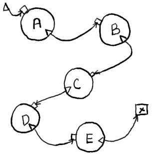
List

- $\mathcal{L}ist$ sends a set A to the set of 'lists' of elements in A

$$\mathcal{L}ist : \mathbf{Set} \to \mathbf{Set}$$
$$\mathcal{L}ist(A) = \{()\} \cup \{(x, xlist) | x \in A, xlist \in \mathcal{L}ist(A)\}$$

- $()$ is called the empty list

$$(1, (2, (3, (4, ())))) \in \mathcal{L}ist(\mathbb{Z})$$
$$(1/2, (Nothing, (1/4, ()))) \in \mathcal{L}ist(\mathcal{M}aybe(\mathbb{Q}))$$
$$(1, 2, 3, 4) \in \mathcal{L}ist(\mathbb{Z})$$



Linked List

List

- $\mathcal{L}ist$ is an endofunctor on **Set**
- Needs a mapping for the morphisms (functions)

$$\mathcal{L}map : Hom(A, B) \to Hom(\mathcal{L}ist(A), \mathcal{L}ist(B))$$
$$\mathcal{L}map(f)(()) = ()$$
$$\mathcal{L}map(f)((x, xlist)) = (f(x), \mathcal{L}map(f)(xlist))$$

For $f(x) = x^2$, $\mathcal{L}map(f)((1, 2, 3, 4)) = (1, 4, 9, 16)$

- Clearly satisfies functor laws (identity and composition)

List

- $\mathcal{L}ist$ is an endofunctor on **Set**
- Needs a mapping for the morphisms (functions)

$$\mathcal{L}map : Hom(A, B) \to Hom(\mathcal{L}ist(A), \mathcal{L}ist(B))$$
$$\mathcal{L}map(f)(()) = ()$$
$$\mathcal{L}map(f)((x, xlist)) = (f(x), \mathcal{L}map(f)(xlist))$$

For $f(x) = x^2$, $\mathcal{L}map(f)((1, 2, 3, 4)) = (1, 4, 9, 16)$

- Clearly satisfies functor laws (identity and composition)

List

- $\mathcal{L}ist$ is an endofunctor on **Set**
- Needs a mapping for the morphisms (functions)

$$\mathcal{L}map : Hom(A, B) \to Hom(\mathcal{L}ist(A), \mathcal{L}ist(B))$$
$$\mathcal{L}map(f)(()) = ()$$
$$\mathcal{L}map(f)((x, xlist)) = (f(x), \mathcal{L}map(f)(xlist))$$

For $f(x) = x^2$, $\mathcal{L}map(f)((1, 2, 3, 4)) = (1, 4, 9, 16)$

- Clearly satisfies functor laws (identity and composition)

List

- $\mathcal{L}ist$ is an endofunctor on **Set**
- Needs a mapping for the morphisms (functions)

$$\mathcal{L}map : Hom(A, B) \to Hom(\mathcal{L}ist(A), \mathcal{L}ist(B))$$
$$\mathcal{L}map(f)(()) = ()$$
$$\mathcal{L}map(f)((x, xlist)) = (f(x), \mathcal{L}map(f)(xlist))$$

For $f(x) = x^2$, $\mathcal{L}map(f)((1, 2, 3, 4)) = (1, 4, 9, 16)$

- Clearly satisfies functor laws (identity and composition)

- What does an endofunctor on **Set** to do a set of functions?
- An applicative functor is a functor with a 'splat' function

$$\mathcal{F}splat : \mathcal{F}(Hom(A, B)) \to Hom(\mathcal{F}(A), \mathcal{F}(B))$$

- $\mathcal{F}splat$ can also be defined as a binary function

$$\mathcal{F}splat : \mathcal{F}(Hom(A, B)) \times \mathcal{F}(A) \to \mathcal{F}(B)$$

- There are rules applicative functors must follow

Applicative Functors

- What does an endofunctor on **Set** to do a set of functions?
- An applicative functor is a functor with a ‘splat’ function

$$\mathcal{F}splat : \mathcal{F}(Hom(A, B)) \to Hom(\mathcal{F}(A), \mathcal{F}(B))$$

- $\mathcal{F}splat$ can also be defined as a binary function

$$\mathcal{F}splat : \mathcal{F}(Hom(A, B)) \times \mathcal{F}(A) \to \mathcal{F}(B)$$

- There are rules applicative functors must follow

Applicative Functors

- What does an endofunctor on **Set** to do a set of functions?
- An applicative functor is a functor with a 'splat' function

$$\mathcal{F}splat : \mathcal{F}(Hom(A, B)) \to Hom(\mathcal{F}(A), \mathcal{F}(B))$$

- $\mathcal{F}splat$ can also be defined as a binary function

$$\mathcal{F}splat : \mathcal{F}(Hom(A, B)) \times \mathcal{F}(A) \to \mathcal{F}(B)$$

- There are rules applicative functors must follow

Applicative Functors

- What does an endofunctor on **Set** to do a set of functions?
- An applicative functor is a functor with a 'splat' function

$$\mathcal{F}splat : \mathcal{F}(Hom(A, B)) \to Hom(\mathcal{F}(A), \mathcal{F}(B))$$

- $\mathcal{F}splat$ can also be defined as a binary function

$$\mathcal{F}splat : \mathcal{F}(Hom(A, B)) \times \mathcal{F}(A) \to \mathcal{F}(B)$$

- There are rules applicative functors must follow

Applicative Functors

- $\mathcal{M}aybe$ is an applicative functor

$$\mathcal{M}splat : \mathcal{M}aybe(Hom(A,B)) \times \mathcal{M}aybe(A) \to \mathcal{M}aybe(B)$$
$$\mathcal{M}splat(Nothing)(\_) = \mathcal{M}splat(\_)(Nothing) = Nothing$$
$$\mathcal{M}splat(f)(x) = f(x)$$

- $\mathcal{L}ist$ is an applicative functor

$$\mathcal{L}splat : \mathcal{L}ist(Hom(A,B)) \times \mathcal{L}ist(A) \to \mathcal{L}ist(B)$$
$$\mathcal{L}splat_1(())(\_) = \mathcal{L}splat_1(\_)(()) = ()$$
$$\mathcal{L}splat_1((f, flist))((x, xlist)) = (f(x), \mathcal{L}splat_1(flist)(xlist))$$

- Could $\mathcal{L}ist$ be an applicative functor in any other ways?

Applicative Functors

- $\mathcal{M}aybe$ is an applicative functor

$$\mathcal{M}splat : \mathcal{M}aybe(Hom(A,B)) \times \mathcal{M}aybe(A) \to \mathcal{M}aybe(B)$$
$$\mathcal{M}splat(Nothing)(\_) = \mathcal{M}splat(\_)(Nothing) = Nothing$$
$$\mathcal{M}splat(f)(x) = f(x)$$

- $\mathcal{L}ist$ is an applicative functor

$$\mathcal{L}splat : \mathcal{L}ist(Hom(A,B)) \times \mathcal{L}ist(A) \to \mathcal{L}ist(B)$$
$$\mathcal{L}splat_1(())(\_) = \mathcal{L}splat_1(\_)(()) = ()$$
$$\mathcal{L}splat_1((f, flist))((x, xlist)) = (f(x), \mathcal{L}splat_1(flist)(xlist))$$

- Could $\mathcal{L}ist$ be an applicative functor in any other ways?

Applicative Functors

- $\mathcal{M}aybe$ is an applicative functor

$$\mathcal{M}splat : \mathcal{M}aybe(Hom(A, B)) \times \mathcal{M}aybe(A) \to \mathcal{M}aybe(B)$$
$$\mathcal{M}splat(Nothing)(\_) = \mathcal{M}splat(\_)(Nothing) = Nothing$$
$$\mathcal{M}splat(f)(x) = f(x)$$

- $\mathcal{L}ist$ is an applicative functor

$$\mathcal{L}splat : \mathcal{L}ist(Hom(A, B)) \times \mathcal{L}ist(A) \to \mathcal{L}ist(B)$$
$$\mathcal{L}splat_1(())(\_) = \mathcal{L}splat_1(\_)(()) = ()$$
$$\mathcal{L}splat_1((f, flist))((x, xlist)) = (f(x), \mathcal{L}splat_1(flist)(xlist))$$

- Could $\mathcal{L}ist$ be an applicative functor in any other ways?

Sources & Images

Sources

- Abstract Algebra by Dummit and Foote
- http://en.wikibooks.org/wiki/Haskell/Category_theory
- Lectures by and conversations with Kenny Foner

Images

- https://bartoszmilewski.files.wordpress.com/2014/10/img_1330.jp
- http://shuklan.com/haskell/L12_files/category.png
- http://ncatlab.org/nlab/files/functor.jpg
- http://www.programmingtunes.com/wp-content/uploads/2013/04/Types-of-Data.jpg
- https://lh3.googleusercontent.com/proxy/W-kz6vWx9ntZrS2FCduApSQ0E-YsddspOrfnWyKP2J-49Uu8_5ahu-lOEfHLmT7w2lZMvQ_vhDGxCkqHlMo1C_0VCkrCFeSzfvtW4PjD